PB80-177124

Coal Preparation Plant Computer Model
Volume II. Program Documentation

Battelle Columbus Labs., OH

Prepared for

Industrial Environmental Research Lab.
Research Triangle Park, NC

Jan 80

U.S. Department of Commerce
National Technical Information Service
NTIS

United States
Environmental Protection
Agency

Industrial Environmental Research
Laboratory
Research Triangle Park NC 27711

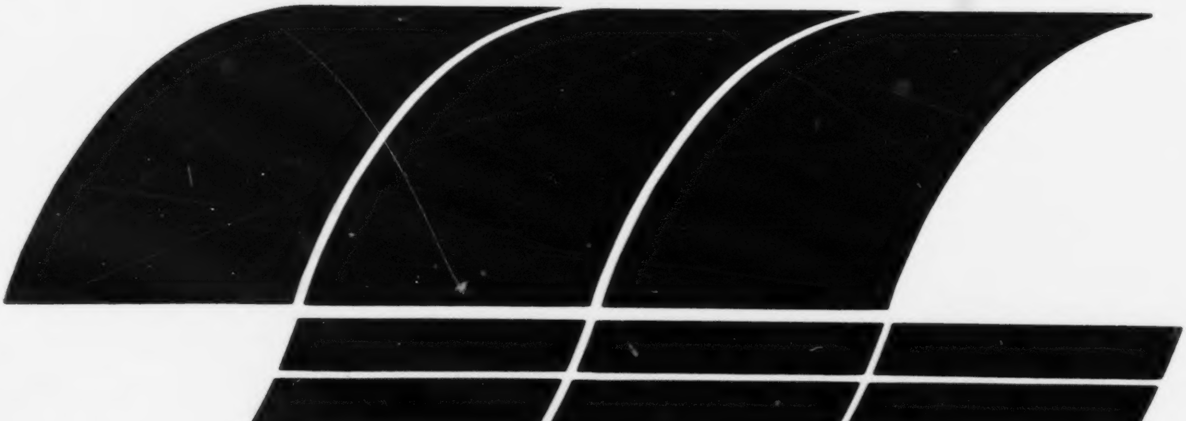
EPA-600/7-80-010b
January 1980

PB80-177124

EPA

Coal Preparation Plant Computer Model:
Volume II.
Program Documentation

Interagency Energy/Environment R&D Program Report



REPRODUCED BY
NATIONAL TECHNICAL
INFORMATION SERVICE
U. S. DEPARTMENT OF COMMERCE
SPRINGFIELD, VA. 22161

RESEARCH REPORTING SERIES

Research reports of the Office of Research and Development, U.S. Environmental Protection Agency, have been grouped into nine series. These nine broad categories were established to facilitate further development and application of environmental technology. Elimination of traditional grouping was consciously planned to foster technology transfer and a maximum interface in related fields. The nine series are:

1. Environmental Health Effects Research
2. Environmental Protection Technology
3. Ecological Research
4. Environmental Monitoring
5. Socioeconomic Environmental Studies
6. Scientific and Technical Assessment Reports (STAR)
7. Interagency Energy-Environment Research and Development
8. "Special" Reports
9. Miscellaneous Reports

This report has been assigned to the INTERAGENCY ENERGY-ENVIRONMENT RESEARCH AND DEVELOPMENT series. Reports in this series result from the effort funded under the 17-agency Federal Energy/Environment Research and Development Program. These studies relate to EPA's mission to protect the public health and welfare from adverse effects of pollutants associated with energy systems. The goal of the Program is to assure the rapid development of domestic energy supplies in an environmentally-compatible manner by providing the necessary environmental data and control technology. Investigations include analyses of the transport of energy-related pollutants and their health and ecological effects; assessments of, and development of, control technologies for energy systems; and integrated assessments of a wide range of energy-related environmental issues.

EPA REVIEW NOTICE

This report has been reviewed by the participating Federal Agencies, and approved for publication. Approval does not signify that the contents necessarily reflect the views and policies of the Government, nor does mention of trade names or commercial products constitute endorsement or recommendation for use.

This document is available to the public through the National Technical Information Service, Springfield, Virginia 22161.

TECHNICAL REPORT DATA

(Please read Instructions on the reverse before completing)

1. REPORT NO. EPA-600/7-80-010b	2.	3. RECIPIENT'S ACCESSION NO. PB80-177124
4. TITLE AND SUBTITLE Coal Preparation Plant Computer Model: Volume II. Program Documentation		5. REPORT DATE January 1980 6. PERFORMING ORGANIZATION CODE
7. AUTHOR(S) Frederick K. Goodman and Jane H. McCreery		8. PERFORMING ORGANIZATION REPORT NO.
9. PERFORMING ORGANIZATION NAME AND ADDRESS Battelle Memorial Institute Columbus Laboratories 505 King Avenue Columbus, Ohio 43201		10. PROGRAM ELEMENT NO. EHE623A 11. CONTRACT/GRANT NO. 68-02-2163, Task 814
12. SPONSORING AGENCY NAME AND ADDRESS EPA, Office of Research and Development Industrial Environmental Research Laboratory Research Triangle Park, NC 27711		13. TYPE OF REPORT AND PERIOD COVERED Task Final; 7/76 - 7/79 14. SPONSORING AGENCY CODE EPA/600/13

15. SUPPLEMENTARY NOTES IERL-RTP project officer is James D. Kilgroe, Mail Drop 61, 919/541-2851.

16. ABSTRACT The two-volume report describes a steady state modeling system that simulates the performance of coal preparation plants. The system was developed originally under the technical leadership of the U.S. Bureau of Mines and the sponsorship of the EPA. The modified form described in this report, written in FORTRAN, was developed by Battelle for the EPA. The original modifications made the program usable in evaluating an advanced coal cleaning facility being constructed at Homer City, PA. Subsequent changes allowed the model to be used for a wider range of performance and cost evaluations. Initial changes to the original program increased the number of process operations which could be simulated, and simplified program operation. Later modifications permitted the calculation of plant water flows and the estimation of plant costs. Volume I contains user documentation, and Volume II provides process documentation. Volume I describes: the manner in which coal flows are represented, the mathematical approach of the various unit operations, the cost evaluation approach, preparation of the input, and interpretation of the output (the last two in terms of an example). Program documentation begins in Volume II with a discussion of basic documentation principles, followed by presentation of each routine and common block in terms of these principles.

17. KEY WORDS AND DOCUMENT ANALYSIS

a. DESCRIPTORS		b. IDENTIFIERS/OPEN ENDED TERMS	c. COSATI Field/Group	
Pollution	Cost Estimates	Pollution Control	13B	14A
Coal		Stationary Sources	08G	
Coal Preparation		Coal Cleaning	08I	
Mathematical Models			12A	
FORTRAN			09A	
Performance Evaluation			05A	

18. DISTRIBUTION STATEMENT Release to Public	19. SECURITY CLASS *(This Report)* Unclassified	21. NO. OF PAGES 316
	20. SECURITY CLASS *(This page)* Unclassified	22. PRICE

Reproduced from best available copy.

EPA Form 2220-1 (9-73)

EPA-600/7-80-010b

January 1980

Coal Preparation Plant Computer Model:
Volume II.
Program Documentation

by

Frederick K. Goodman and Jane H. McCreery

Battelle Memorial Institute
Columbus Laboratories
505 King Avenue
Columbus, Ohio 43201

Contract No. 68-02-2163
Task No. 814
Program Element No. EHE623A

EPA Project Officer: James D. Kilgroe

Industrial Environmental Research Laboratory
Office of Environmental Engineering and Technology
Research Triangle Park, NC 27711

1-A

Prepared for

U.S. ENVIRONMENTAL PROTECTION AGENCY
Office of Research and Development
Washington, DC 20460

FOREWORD

Many elements and chemical compounds are known to be toxic to man and other biological species. But our knowledge concerning the levels and conditions under which these substances are toxic is extremely limited. Further, little is known concerning the emission of these pollutants from industrial processes and the mechanism by which they are transported, transformed, dispersed, or accumulated in our environment.

Portions of the Federal Clean Air Act, the Resource Conservation Recovery Act, and the Federal Water Pollution Control Act require the U.S. Environmental Protection Agency (EPA) to identify and regulate hazardous or toxic substances which result from man's industrial activities. Industrial pollutants are often identified only after harmful health or ecological effects are noted. Remedial actions are costly, the damage to human and other biological populations is often irreversible, and the persistence of some environmental contaminants may endanger future populations.

EPA's Office of Research and Development is responsible for health and ecological research, studies concerning the transportation and fate of pollutants, and the development of technologies for controlling industrial pollutants. As a part of this office, the Industrial Environmental Research Laboratory, which is responsible for development of pollution control technology, conducts a large environmental assessment program. The primary objectives of this program are:

- The development of information on the quantities of toxic pollutants emitted from various industrial processes--information needed to prioritize health and ecological research efforts.

- The identification of industrial pollutant emissions which pose a clearly evident health or ecological risk and which should be regulated.

- The evaluation and development of technologies for controlling pollution from these toxic substances.

The coal cleaning environmental assessment program has as its specific objectives the evaluation of pollution and pollution control problems which are unique to coal preparation, storage, and transportation. The coal preparation industry is a mature yet changing industry, and in recent years significant achievements have been made in pollution abatement. As an aid in the evaluation of an advanced physical coal cleaning facility being constructed near Homer City, Pennsylvania, an existing computer program was verified and modified to simulate the design and operation of physical coal preparation plants. Specifically, this report provides user documentation (Volume I) and programmer documentation (Volume II) for Coal Preparation Simulation Model Version 4 (CPSM4) as modified by Battelle.

PREFACE

Since July, 1976, Battelle's Columbus Laboratories has been conducting a major study for the U.S. EPA on the environmental assessment of coal cleaning processes. In one portion of this study, existing computer programs capable of modeling coal preparation plants were verified and modified to aid in the evaluation of an advanced coal cleaning facility being constructed at Homer City, Pennsylvania. The only computer program found with the required capabilities was one developed in FORTRAN under the technical leadership of the U.S. Bureau of Mines and the sponsorship of the U.S. EPA. The original version of this program, Coal Preparation Simulation Model Version 4 (CPSM4) is described in Gottfried, Jacobsen, and Vaillant(1)*.

This document describes in detail a Battelle modified program CPSM4. The Battelle version of CPSM4 maintains all of the basic capabilities of the original version; however, these have been restructured and augmented extensively to make the program usable in the evaluation of a particular plant.

The actual presentation is divided into two volumes: the first contains user documentation and the second contains program documentation. The goal of the first volume is to make the program a usable and understandable tool for the nonprogrammer interested in the simulation of coal preparation plants. The second volume is intended for individuals with knowledge of the FORTRAN language who are interested in the detailed implementation of the material presented in Volume I.

---

* Reference numbers refer to the Reference List at the end of Volume I. A separate reference list is included at the end of Volume II.

Although pages in each volume are numbered sequentially, tables and figures are numbered by volume, chapter, and part followed by a sequence number as shown below. The numbering of equations begins at (1) within each part.

```
II-12-6.2
|   |  | |
|   |  | sequence number
|   |  |
|   |  part
|   |
|   chapter
|
volume
```

ABSTRACT

This report documents a steady state modeling system that simulates the performance of coal preparation plants. This modeling system was developed originally under the technical leadership of the U.S. Bureau of Mines and the sponsorship of the U.S. Environmental Protection Agency. The modified form described in this report was developed by Battelle for the U.S. Environmental Protection Agency. The system is written in FORTRAN.

The original purpose of the modifications was to make the program usable in the evaluation of an advanced coal cleaning facility being constructed at Homer City, Pennsylvania. Subsequent changes were made to allow the model to be used for a wider range of performance and cost evaluations. Initial changes to the original program were made to (1) increase the number of process operations which could be similated and (2) simplify operation of the program. Later modifications were made to permit the calculation of plant water flows and the estimation of plant costs.

The report is divided into two volumes; Volume I contains user documentation, and Volume II provides program documentation. The manner in which flows of coal are represented and the mathematical approach of the various unit operations are discussed in detail in Volume I. Also, the approach taken for the evaluation of costs is presented. The preparation of the input and the interpretation of the output are described in terms of an example. In Volume II, the program documentation begins with a discussion of the basic principles of documentation and then presents each routine and common block in the system in terms of these principles.

This report covers work during the period from July 1976 through July 1979.

TABLE OF CONTENTS
(Volume II)

Page

FOREWORD ii
PREFACE iii
ABSTRACT v
LIST OF FIGURES viii
ACKNOWLEDGEMENTS xi

1.0 INTRODUCTION 1

  1.1 Basic Principles 4

    1.1.1 Machine Independence 4
    1.1.2 Localizing Machine Dependence 6
    1.1.3 Localizing Single Decisions 6

  1.2 Documentation Conventions 8

2.0 INFORMATION STORAGE 13

  2.1 Dynamic Storage Allocation 14
  2.2 The /WORK/ Common Block 17
  2.3 The /CNTVAR/ Common Block 21
  2.4 The /LOCALV/ Common Block 23

3.0 OVERALL STRUCTURE 24

  3.1 The BLOCK DATA Subroutine 27
  3.2 Program CPSM4 29
  3.3 Subroutine SIMUL 35

4.0 COMPUTATIONAL UTILITIES 51

  4.1 Subroutine CMPBRK 52
  4.2 Subroutine BRSIZE 56
  4.3 Interpolation and Approximation 67
  4.4 Subroutine INTS 69
  4.5 Function YINTRP 73
  4.6 Subroutine SDSIZE 79
  4.7 Subroutine DSTRB 82
  4.8 Subroutine ECON 86

5.0 THE UNIT OPERATIONS 98

  5.1 Subroutine BREAK2 99
  5.2 Subroutine NOTPTH 104
  5.3 Subroutine SCREEN 108
  5.4 Subroutine DCWASH 112
  5.5 Subroutine WASH 119
  5.6 Subroutine WASH2 122

TABLE OF CONTENTS
(Continued)

Page

5.7 Subroutine FROTH 126
5.8 Subroutine BREAK 135
5.9 Subroutine ROTARY 140
5.10 Subroutine CRUSH 147

6.0 CHARACTER MANIPULATION 156

6.1 Displaying Size Information 158
6.2 Displaying Unit Descriptors 160
6.3 Internal Character Display Codes 161
6.4 The /CHRSET/ Common Block 169
6.5 The /CODERC/ Common Block 172
6.6 The /FILES/ Common Block 173
6.7 The /CONST/ Common Block 174
6.8 Subroutine CNVTIE 175
6.9 Subroutine PRSIZE 182
6.10 Subroutine RDDESC 191
6.11 Subroutine PRDESC 195
6.12 Subractive RDVAL 198

7.0 INPUT PROCESSING 201

7.1 Subroutine RDUNIT 202
7.2 Subroutine RDCURV 206
7.3 Subroutine RDFEED 214
7.4 Subroutine CONFSV 219
7.5 Subroutine FEEDSV 222
7.6 Subroutine RDCFBS 226
7.7 Subroutine RDFDBS 231
7.8 Subroutine SVBASE 236
7.9 Subroutine RDBASE 239

8.0 REPORT GENERATION 243

8.1 The /BLK1/ and /BLK2/ Common Blocks 244
8.2 Subroutine VESSEL 247
8.3 Subroutine PRHEAD 250
8.4 Subroutine OUTPT1 253
8.5 Subroutine CMPUS 264
8.6 Subroutine OUTPT2 273
8.7 Subroutine OUTPT4 288
8.8 Subroutine PRLINE 299
8.9 The /IOCNT/ Common Block 302
8.10 The /FLAGS/ Common Block 303

9.0 REFERENCE LIST FOR VOLUME II 304

LIST OF FIGURES

Figure No. Page

II-1-1.1 A Machine-Independent ANSI FORTRAN Program 5

II-1-2.1 Flowcharting Symbols 12

II-2-2.1 Equipment Information Control Block 19

II-2-2.2 Curve Set Information Block 20

II-2-3.1 Content of /CNTVAR/ Common Block 22

II-3-0.1 Elements of Program 25

II-3-1.1 Listing of Block Data Subroutine 28

II-3-2.1 Listing of Program CPSM4 30

II-3-3.1 Listing of Subroutine SIMUL 36

II-4-1.1 Listing of Subroutine CMPBRK 53

II-4-2.1 Percent Ash by Size 57

II-4-2.2 Flowchart for Subroutine BRSIZE 62

II-4-2.3 Listing for Subroutine BRSIZE 63

II-4-4.1 Flowchart for Subroutine INTS 70

II-4-4.2 FORTRAN Listing for Subroutine INTS 71

II-4-5.1 Flowchart for Function YINTRP 74

II-4-5.2 FORTRAN Listing for YINTRP 75

II-4-6.1 FORTRAN Listing for Subroutine SDSIZE 80

II-4-7.1 Listing of Subroutine DSTRB 83

II-4-8.1 Listing of Subroutine ECON 87

II-5-1.1 Listing for Subroutine BREAK2 100

II-5-2.1 Listing for Subroutine NOTPTH 105

II-5-3.1 Listing for Subroutine SCREEN 109

II-5-4.1 Listing for Subroutine DCWASH 113

LIST OF FIGURES
(Continued)

Figure No. Page

II-5-5.1 Listing for Subroutine WASH 120

II-5-6.1 Listing for Subroutine WASH2 123

II-5-7.1 Listing for Subroutine FROTH 127

II-5-8.1 Listing for Subroutine BREAK 136

II-5-9.1 Listing for Subroutine ROTARY 141

II-5-10.1 Listing for Subroutine CRUSH 148

II-6-3.1 CDC Numeric Internal Display Codes 166

II-6-3.2 Simple Numeric Conversion Routine 166

II-6-3.3 Function IRJLZ 167

II-6-4.1 Initialization of Letter Array 169

II-6-4.2 Internal Display Codes for CPSM4 171

II-6-6.1 Initialization of /FILES/ Variables 173

II-6-7.1 Initialization of /CONST/ Variables 174

II-6-8.1 Listing for Subroutine CNVTIE 176

II-6-9.1 Mesh Sizes 183

II-6-9.2 Fractional Sizes 184

II-6-9.3 Flowchart for Subroutine PRSIZE 186

II-6-9.4 FORTRAN Listing for Subroutine PRSIZE 187

II-6-10.1 FORTRAN Listing for Subroutine RDDESC 192

II-6-11.1 FORTRAN Listing for Subroutine PRDESC 196

II-6-12.1 Listing for Subroutine RDVAL 199

LIST OF FIGURES
(Continued)

Figure No.		Page
II-7-1.1	Listing for Subroutine RDUNIT	203
II-7-2.1	Listing for Subroutine RDCURV	207
II-7-3.1	Listing for Subroutine RDFEED	215
II-7-4.1	Listing for Subroutine CONFSV	220
II-7-5.1	Listing for Subroutine FEEDSV	223
II-7-6.1	Listing for Subroutine RDCFBS	227
II-7-7.1	Listing for Subroutine RDFDBS	232
II-7-8.1	Listing for Subroutine SVBASE	237
II-7-9.1	Listing for Subroutine RDBASE	240
II-8-1.1	Contents of /BLK1/ Common Block	245
II-8-2.1	Listing for Subroutine VESSEL	248
II-8-3.1	Listing for Subroutine PRHEAD	251
II-8-4.1	Listing for Subroutine OUTPT1	254
II-8-5.1	Listing for Subroutine CMPUS	265
II-8-6.1	Listing for Subroutine OUTPT2	274
II-8-7.1	Listing for Subroutine OUTPT4	289
II-8-8.1	Listing for Subroutine PRLINE	300

ACKNOWLEDGEMENTS

This study was conducted as a Task on Battelle's Columbus Laboratories' program "Environmental Assessment of Coal Cleaning Processes" supported by the Industrial Environmental Research Laboratory, U.S. Environmental Protection Agency, Research Triangle Park, North Carolina.

Dr. Byron S. Gottfried, Departments of Industrial Engineering and Energy Resources at the University of Pittsburgh, and P. Stanley Jacobsen, Coal Preparation and Analysis Laboratory of the U.S. Bureau of Mines, are thanked for their patient assistance. Valuable assistance has been provided, through comments in several progress review meetings, by Mr. Ed Zawadzki, consultant to General Public Utilities Corporation, Messrs. Charles Statler, Ray McGraw, and Jim Tice of Pennsylvania Electric Company, Dr. Gerald Janik of New York State Electric and Gas Corporation, and Mr. Ken Harrison of Heyl and Patterson, Inc.

The contributions of Mr. G. Ray Smithson, Jr., Program Manager, Mr. Alexis W. Lemmon, Jr., Deputy Program Manager, and Dr. Gerald L. Robinson, Task Leader, are gratefully acknowledged.

The advice, counsel, and comments of the EPA Project Officer, Mr. James D. Kilgroe, and others at the IERL/RTP facility were invaluable in performance of this work.

II PROGRAM DOCUMENTATION

1.0 INTRODUCTION

The documentation of large computer programs is never easy. Theoretically, it has a simple goal - to describe exactly how the program does what it does. Very generally, the documentation has two primary purposes. From the standpoint of the final user of the program, it tells him how to use the program and explains exactly how the results are derived. This information is needed by the final user, if he is to use the program effectively and if he is to evaluate its results. From the standpoint of the programmer who must maintain, debug, alter, or enhance the program, the documentation is needed so that he can fully understand the programming conventions and algorithms used. It is this second type of documentation, programmer's documentation, which causes the most difficulty. Throughout the remainder of this work, the term documentation will be used to refer to that documentation written for the use of programmers.

For simple programs the documentation consists of a series of descriptions and flowcharts which describe the program. This description includes a brief presentation of each routine and common block, a description of the variables used, and a description of the actual program instructions and their relationships. The purpose of such documentation, then, is to describe the program, where the term program refers to the actual set of instructions which make it up. The production of this type of documentation is straightforward and often quite mechanical. Despite the tediousness of its production, this level of basic documentation is important and must be made available.

As programs get larger, however, the above level of basic documentation becomes increasingly less than satisfying and sufficient. Before the specific programming code description can be useful to a programmer who is considering making some change, he must first understand the approach being taken.

He must first fully understand what a particular section of code is intended to do before he can hope to appreciate a description of the particular logic used to do it. A second level of documentation is then needed. This second level describes what the larger sections of code do and how these sections are logically interrelated. This second level represents the functional documentation of the major components in the system.

As an example of the distinction between functional documentation and basic documentation, consider any one of the examples in Carnahan, et al.(1)*, on the use of applied numerical methods. For example they show the application of a fourth order Runga-kutta method to the transient behavior of a resonant circuit. Pages 370 through 375 include flowcharts of the routines used, descriptions of the variables, brief descriptions of the routines, and the FORTRAN listing of those routines. These pages are an excellent example of basic documentation.

Suppose now that the above five pages alone were copied and given to a programmer along with a sample run and the complaint that the results seemed incorrect. Could he fix the program or at least diagnose the problem? The answer would almost certainly be that he could not. Additional information would be needed about the Runga-kutta approach to the solution of differential equations and about the actual mathematical form of the circuit problem. Such information is, of course, provided in the remainder of Chapter 6 of Carnahan et al. This chapter is an excellent example of functional documentation. Unfortunately, there is still a real potential problem. What if the programmer selected did not happen to know about differential equations, or perhaps knew no calculus at all? It is this sort of problem that is often faced in attempts to document large computer programs such as CPSM4.

Before the algorithms and functional relationships used in a program can be appreciated and understood by some programmer, he must understand the methodology or basic approach from which they derive. Why were the particular set of functional relationships chosen to solve the problem at hand? What were the alternative approaches and why were they rejected? Questions such as these relate to the methodology used in the design of the computer program being documented. It is to this level of methodological documentation that much of the discussion in this presentation belongs.

---

* Reference numbers in this volume refer to the Reference List on page 304.

Methodological documentation differs from basic and functional documentation in one very important way. It need not be specific to any particular computer program. It does not describe how a particular program is written; rather, it provides the information needed to understand why a possibly large group of computer programs are written the way they are. The above is not, however, intended to imply that no actual computer programs will be presented. The opposite is true. Program CPSM4, along with a number of examples, will be explicitly described. This description will also include, however, the methodological underpinnings of the program.

1.1 Basic Principles

Throughout the ensuing discussion many opinions are expressed upon which conclusions are drawn and programs will be written. It is the author's hope that his opinions relative to this area are self-consistent and reasonable. They are based on a series of general, interrelated, basic principles. Though these are premises - i.e., cannot be proved and/or have not been empirically verified - they nontheless seem true and can be stated explicitly though not formally. This part of the chapter introduces them as a group. Each will be discussed and used in more detail in various places throughout this documentation.

1.1.1 Machine Independence

Programs, in this case FORTRAN programs, are often measured in terms of their machine independence. Without becoming overly pedantic, the term machine independence seems to have at least two uses. In one sense, it is used to measure the likelihood that the program will run on any randomly selected computer without requiring major programming changes. A program is said to be machine independent if it has a high likelihood of running on many different machines. It is, in essence, in this sense that the term will be used here. There is, however, another use of the term. It is used to measure that proportion of the program which is written in ANSI FORTRAN. Since every major FORTRAN compiler must include ANSI FORTRAN as part of its language interface, the assumption in the above use of the term machine independence is that if a program is written in ANSI FORTRAN, then it will be usable on any machine having a FORTRAN compiler.

Machine independence, in the first sense of the term, is not just a syntax problem if FORTRAN is being used as the programming language. The fact that a program is written in such a way that it will be compiled successfully by any FORTRAN compiler, does not imply that it will execute in the same way on each computer. ANSI FORTRAN is not semantically a machine-independent language. This fact is especially true of the FORMAT statement. As an example of the above lack of semantic machine independence on the part of

ANSI FORTRAN, consider the extremely simple problem shown in Figure II-1-1.1. The problem here is to write a computer program that will first read a name, containing no more than 20 characters and punched on a card beginning in column 1, and then write it introduced by the message "the name is".

This program would compile under any FORTRAN compiler, but it would produce the desired results only on a CDC 6000 or 7000 series computer. The reason is simple. Only the above CDC series computers have word size and internal character display codes constructed to allow storage of 10 characters per computer word. Any other computer would ignore characters within the fields defined and would not, therefore, produce the desired results.

```
   DIMENSION NAME (2)
1  FORMAT(2A10)
2  FORMAT(1H1,10X,11HTHE NAME IS,1X,2A10)
   READ(5,1)(NAME(I),I=1,2)
   WRITE(6,2)(NAME(I),I=1,2)
   STOP
   END
```

FIGURE II-1-1.1. A MACHINE-INDEPENDENT ANSI FORTRAN PROGRAM

The point to be recognized here is that the results produced by a FORTRAN program are not just dependent on the syntax of that program. They are also dependent upon the internal characteristics of the computer compiling and executing the program. Putting this more precisely, it is the semantics of a FORTRAN program that finally determine its independence, given that it has syntactic independence. Being written in ANSI FORTRAN is a necessary but not sufficient condition for a program to be machine independent.

1.1.2 Localizing Machine Dependence

It must be emphasized that few significant programs including program CPSM4 will be completely machine independent - i.e., will run without any change whatsoever on any computer with an ANSI FORTRAN compiler. It is the responsibility, however, of the programmer to minimize the machine dependencies whenever possible if he is claiming to write machine-independent code. Furthermore, those dependencies should be localized as much as possible in the code. It is an absolute law that the probability of error increases significantly as the number of changes needed in a program increases and as those changes become more dispersed. If a program must be changed in order to move it to another machine, then it is the programmer's responsibility to minimize the probability of error associated with that move.

1.1.3 Localizing Single Decisions

Closely related to the notion of localizing machine dependencies, is the principle of localizing single decisions. This methodological principle, stated very simply, is that whenever possible a single decision on the part of the programmer should be reflected or implemented within a single block of statements in the language interface. There are numerous examples of this principle in the following presentation.

The use of this principle has two obvious benefits. First, because the decision has been implemented at one local point, it will be easy to find in the program and easy to change when necessary without introducing error. The localization also means that the making of that decision will be explicitly discussed in the basic documentation describing that area of the code. As a result, the program will be easier for others to understand and work with.

The second benefit that derives from using the principle of localizing single decisions is that it encourages the programmer to think about the problem that he is programming more deeply than he might otherwise. It is very easy to program every problem as though it consisted of a collection of special cases each of which is independent of every other. Such programming, though it often tends to execute efficiently from the standpoint of absolute central processing time, is highly undesirable from the standpoint of large program design for the very simple reason that it generates programs that become too long to work with effectively. There is much to be said for economy of notation in programming and the principle of localizing and, therefore, finding single decisions helps greatly in achieving the economy.

1.2 Documentation Conventions

In the following discussion, a large number of FORTRAN routines are presented. In general, these routines have one of two purposes. Many are intended to serve only as an example for some particular point that is being made. These are always quite short and often given in fragmentary form. The purpose and function of these routines is obvious from the context of the discussion. No effort is made to document them further. The other set of routines are those that actually make up the program. Each of these routines is documented in detail. This chapter discusses the conventions to be used in this documentation.

The methodological documentation for many of the routines within CPSM4 has already been given in Section I. The material in this section assumes that the reader is familiar with it. In those cases where additional methodological discussion is needed, it is presented in the part immediately preceding the part on the relevant routine. Each routine is given its own part within the presentation. The text of each part contains necessary functional documentation to make the basic documentation clear. The basic documentation is given in the form of one or two figures which present the functional documentation for that routine. These are the flowchart for the routine and the listing of the routine as generated by the FTN FORTRAN compiler on the CDC machine at Battelle's Columbus Laboratories. All basic descriptive documentation for a routine is included in the comments for that routine, including a 'verbal' flowchart. No separate discussion constituting the basic documentation of the routines is provided. Since this convention does not agree with normal practice, a brief description of the commenting conventions used in the listings of the routines is given here.

The basic documentation conventions used in the routines presented here are based on two simple premises. First, the primary user of basic documentation is the programmer who is unfamiliar with the routine being documented and who must make some modification in that routine. This unfamiliarity may derive from the fact that the programmer has never seen the routine in question, or it may derive from the fact that though the programmer has worked with the routine before, or even written the routine, he has forfotten the details of the actual coding conventions used. Remember that basic documentation views the routine as a set of instructions. Its purpose is to describe those instructions. In this context, the basic documentation constitutes the documentation of last resort. The programmer who is interested in what a routine does and generally how it does it will read the functional documentation. He will not read the basic documentation until he must deal with the actual code.

The second basic premise upon which the basic documentation conventions are based is that when a programmer needs to know what the actual code of a routine looks like, the document that he will have easiest access to and that he will be primarily interested in, is the listing of the routine itself. The basic documentation will be turned to only when the programmer has some question or doubt about why some line or block of code is constructed in the way it is. The single most frustrating experience in the author's programming life is to have before him the listing of a routine within some large program; the need to understand why some line of code is, or worse is not, there; and the information that the basic documentation is either logistically unavailable or "in preparation".

What follows from the above two premises seems obvious. The place for the basic documentation is in the listing itself. The two should be inseparable. A routine is not truly even written until the basic documentation is completed. The following information is included in the comments of every routine of importance presented here. First the title of the routine introduced by a comment beginning with the word TITLE in column 7. This comment tells the reader basically what the routine does and/or what the name of the routine stands for. The second comment begins with the routine type and name in column 7, followed by a description of what the routine generally does. This description constitutes a brief synopsis or outline of the functional description. The third comment type, which begins with the word PARAMETER in column 7, is included if and only if the routine has any formal parameters.

For each parameter, a brief description of its use is included. The fourth comment type begins with the word COMMON in column 7. This comment is included if and only if the routine has any variables declared in common blocks within that routine. The purpose of this comment is to specify exactly which of the declared common variables are actually used in the routine. Since common variables are used in many routines, no description of their use is included as this description is not germane to the routine. The fifth comment type, which is keyed by the word EXTERNALS in column 7, is used if and only if the routine makes any external references including FORTRAN library routines. Finally, the sixth introductory comment begins with the word VARIABLE in column 7. This comment describes all the local variables used in the routine. All of the comment types are included between the statement introducing the routine and the next named comment card. The basic purpose of this set of comments is straightforward - to define every symbol that appears in an executable statement in the routine.

The executable statements in the routine are broken into a series of steps. Though more than one physical line of code may participate in a single step, the following general rules are followed for breaking a routine into steps. First, all executable statements, with the possible exception of a labeled RETURN statement immediately preceding the END statement, are part of some step in the routine. Second, if a block of statements form a single step, then those statements must be physically contiguous. In so far as branching is concerned, it is unrestricted within a given step; however, no statement external to the step may branch to any statement other than the first statement in the step. Groups of steps may be blocked into sections if the routine is long enough to warrant such additional segmentation. If this is done, no statement outside of a section may branch to any statement within the section other than the first one. The reader may note that these conventions tend to force structured programming. Readers unfamiliar with the term structured programming are referred to Dijkstra(2).

The commenting conventions associated with the above structuring of the routines are as follows. All of the steps in the routine, or within a section if sections are used, are numbered sequentially from one. Each step is introduced with a comment with the word STEP, followed by its number, in column 7. The purpose of the comment is to describe the function of the step and not to describe the form of the statements making up the step. The description of the statements is next to them and there is no need to describe what the reader can see. These comments attempt to deal with the problem that the form of statements often belie their function. The statement itself is the best description of its form. The comment is intended to describe the function associated with that form. An important rule about the content of the step function comment is that if it includes a branch to some other step, or if it includes a reference to some external routine, then that reference must be explicitly mentioned by number or name. The effect of this rule is that the comments tend to form a verbal flowchart of the routine. If sections are included, they too are numbered from one, but they use the Roman numbering system.

In most cases the verbal flowchart formed by the comments is sufficient to show the logical relationships between the statements in the routine being documented. In these cases no additional pictorial flowchart will be included. Occasionally, however, for routines which are particularly complex in so far as branching is concerned, the verbal flowchart is not compact enough to show the structure. In these cases a pictorial flowchart is also provided. Figure II-1-2.1 shows the flowcharting symbols used. The only additional comment to be made is that each step in the routine is explicitly shown in the flowchart via the small numbered circles. This convention simplifies the task of relating the flowchart to the listing.

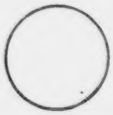
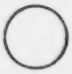
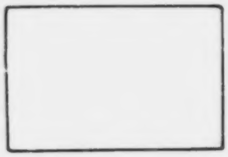
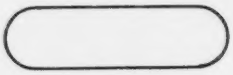
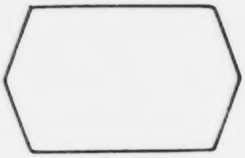
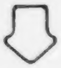
Symbols	Description of use
	Terminal box - indicates beginning or end of a routine.
	Connector box - indicates a step number, as shown in the comments of the actual listing.
	Operation box - contains operations or groups of operations that perform computations.
	Information box - contains information not directly used but helpful in understanding flowchart.
	Decision box - contains statements of a test on which a decision is made.
	Off-page connector.

FIGURE II-1-2.1. FLOWCHARTING SYMBOLS

2.0 INFORMATION STORAGE

Since a coal preparation plant is viewed as a set of units each accepting certain flows as inputs, manipulating those flows, and then producing flows as output, there are basically two types of information that must be stored by the program:

(1) Descriptions of unit operations

(2) Descriptions of flows.

This part describes how this information stored. The first part of this chapter describes the general technique of *dynamic storage allocation* used; while the remaining parts describe how the particular units of information are allocated.

2.1 Dynamic Storage Allocation

A traditional problem in writing FORTRAN programs is providing storage area for the data to be manipulated by the program. There are two ways in which such provisions can be made:

(1) By hard-wiring instructions into the program at the time it is written via explicit DIMENSION statements;

(2) By placing logic into the program that allows it to make provisions as the need arises.

The first technique is referred to as dimensioning the data, and the second as dynamically allocating the data. Program CPSM4 dynamically allocates all equipment specifications and detailed flowstream data.

Dimensioning is usually used in small, application-specific programs. It is easy to program and programs using it are easy to debug. It should never be used, however, in large generalized programs because it places needless restrictions on the user, is extremely inefficient insofar as core utilization is concerned, and makes it very difficult to modify the program.

To use the dimensioning technique, the programmer must make a series of decisions. First, he must decide what types of information are needed. Second, he must decide how much core to allocate to each occurrence of each information type. Third, he must decide how many occurrences of each type should be allowed. Having made all of these decisions, the programmer writes a series of individual DIMENSION statements which reflect all of the above decisions.

As an example of dimensioning shortcomings, an early version of program CPSM4 had the following limitations:

(1) No more than 23 size increments could be used to define the flows

(2) No more than 10 specific gravity increments could be used to define the flows

(3) No more than 13 unit operations could be used to define the plant configuration

(4) No more than 23 flows could be used to define the plant configuration.

In addition, each unit required six cells of information and each flow required four cells of information plus four arrays each dimensioned by size and specific gravity fraction. The total data storage cells then allocated in the program was

$$(6\cdot 13) + 23\cdot(4+4\cdot 23\cdot 10) = 21330.$$

The sample problem which accompanied that early version had 10 units, 19 flows, 8 size fractions, and 8 specific gravity fractions. Had the program been set to these sizes it would have only needed data storage cells equal to

$$(6\cdot 10) + 19\cdot(4+4\cdot 8\cdot 8) = 5000.$$

This sample problem, then, only required 23 percent of the data storage cells which the program had actually provided for. It is in this sense that dimensioning is efficient in so far as core utilization is concerned.

Continuing the example, an early application of the program required using 7 units, 13 flows, 9 size fractions, and 12 specific gravity fractions. This application could not be run using that version of the program because it exceeded the limitation on the number of specific gravity fractions, even though its total data storage requirement was only

$$(6\cdot 7) + 13(4+4\cdot 9\cdot 12) = 5710.$$

It is in this sense that dimensioning places needless restrictions on the user.

The limitation of 10 specific gravity fractions was expressed in every DIMENSION statement in every subroutine and common block manipulating flow data. Using the dimensioning approach, the programmer's decision to limit the maximum number of specific gravity fractions had to be implemented in many different places in the program. Thus, dimensioning forces a violation of the principle of localizing single decisions, discussed earlier. It is in this sense that dimensioning makes it difficult to modify programs.

The alternative is dynamic storage allocation. The disadvantage of dynamic storage allocation is that programs using it must contain additional logic to perform the storage allocations. For large generalized programs, however, this disadvantage is easily outweighed by the fact that this technique places few restrictions on the user; is efficient from the standpoint of core utilization; and greatly reduces the likelihood of program changes necessitated by problems having unexpected data needs.

Basically, dynamic storage allocation is implemented as follows. First the programmer determines the total amount of data storage cells to be allocated. This space is referred to as working storage. Within CPSM4 these are allocated in blank common. At all times the program keeps track of the next available cell in working storage. If at a particular point data storage is needed, it is allocated to the next available cell and the address of the next available cell is incremented by the amount of space needed. The only hard-wired decision that the programmer must make when he writes the program has to do with the overall storage types to be allocated. The details of the implementation are discussed in the following parts of this chapter.

2.2 The /WORK/ Common Block

The /WORK/ common block contains the main control variables needed to allowcate information to the working storage area. The block contains eleven simple integer variables - NEXT, LAST, ICURVE, INUNIT, IDATA, ITITLE, ICHAR, IFLW, IASH, ISUL, and IBTU.

The integer variable NEXT contains the address in working storage of the next available cell. Any routine that requires storage uses the value of this variable as the address at which that storage begins. As storage is used, the variable NEXT is incremented accordingly. As storage is no longer needed, the variable NEXT is decremented. NEXT is initialized at 1 by the BLOCK DATA subroutine.

The integer variable LAST contains the address in working storage of the last available cell. Thus, it specifies the total number of cells available for data storage. If more cells are needed, the program will stop. The condition is detected when the value of NEXT exceeds LAST. LAST is initialized at 6000 by the BLOCK DATA subroutine.

The integer variable ICURVE contains the address in working storage of the start of the equipment type specification. In particular, it contains the address of the first equipment information block. The value of ICURVE is set by the main program. The equipment information block is described in Figure II-2-2.1.

The integer variable IUNIT contains the address in working storage of the start of a vector containing the addresses of the unit descriptions. The value of this variable is set by subroutine RDCFBS.

The integer variable IDATA contains the address in working storage of the start of the detailed flow information. The value of this variable is set by subroutine RDFDBS.

The integer variable ITITLE contains the address in working storage of the start of the run description. The value of this variable is set by the main program.

The integer variable ICHAR contains the address in working storage of the start of the details of the characteristics. The value of this variable is set by subroutine RDFDBS.

The integer variable IFLW contains the address in working storage of the start of the flowstream characteristics. The value of this variable is set by subroutine RDFDBS.

The integer variable IASH contains the sequence number of ash in the characteristics. The value of this variable is supplied by the user.

The integer variable ISUL contains the sequence number of total sulfur in the characteristics. The value of this variable is supplied by the user.

The integer variable IBTU contains the sequence number of the Btu/lb value in the characteristics. The value of this variable is supplied by the user.

Cell	Description of Contents
-1	Equipment type identifier
0	Address of next equipment information block (0 if last block)
1	Address of equipment descriptor
2	Equipment unit operation code

Code	Unit Operation
1	Blending
2	Splitting
3	Screening
4	Single-stage washing
5	Two-stage washing
6	Froth flotation
7	Rotary breaking
8	Crushing

Cell	Description of Contents
3	Address of curve set control blocks (see Figure II-2-2.2)
4+	Miscellaneous constants

Unit Operation	Cell	Constants
Single-stage washing	4	Imperfection denominator offset
Two-stage washing	4	Imperfection denominator offset
Froth flotation	4	Imperfection denominator offset
Rotary breaking	4	Number of falls constant
	5	Hole size threshold
	6	Lower screening constant
	7	Upper screening constant
	8	Height selection factor
Crushing	4	Selection constant
	5	Crushing zone constant
	7	Screening constant

FIGURE II-2-2.1. EQUIPMENT INFORMATION CONTROL BLOCK

Cell	Description of Contents
0	Curve set type code

Code	Meaning
1	One-dimensional vector
2	Univariate distribution
3	Size classification curves
4	c-generalized curves
5	Actual distribution curves

1	Number of points per curve
2	Number of size intervals, curves, or vectors
3	Address of x-points or vectors
4	Address of y-points if relevant
5	Address of special information if relevant

Curve Code	Special Information
3	Size classification decision variable value for each curve
4	Size classification
5	Size classification Number of curves per size Decision variable value for each curve

FIGURE II-2-2.2. CURVE SET INFORMATION BLOCK

2.3 The /CNTVAR/ Common Block

The other common block involved in the storage of unit and flow information is the /CNTVAR/ common block. Figure II-2-3.1 describes the contents of this block.

Variable	Description
KF	This two-dimensional integer array contains the origin and destination unit number for each flow.
SIZE	This one-dimensional real array contains the flowstream size distribution.
GBOUND	This one-dimensional real array contains the specific gravity distribution.
GRAV	This one-dimensional real array contains the mid-points of the specific gravity fractions.
WFLOW	This one-dimensional real array contains the moisture content of each flow.
S	This one-dimensional typeless array contains the flow type codes.
D1	This one-dimensional real array contains the value of the first decision variable for each unit.
D2	This one-dimensional real array contains the value of the second decision variable for each unit.
D3	This one-dimensional real array contains the value of the third decision variable for each unit.
L	This one-dimensional integer array contains the base address in working storage of the equipment information control block pertaining to each unit.
NSIZE	This integer variable contains the number of size fractions in the flowstreams.
NGRAV	This integer variable contains the number of specific gravity fractions in the flowstreams.
IOUT	This integer variable contains the output control code.
NUNITS	This integer variable contains the number of units in the configuration.
NFLOWS	This integer variable contains the number of flows in the configuration.
NCHAR	This integer variable contains the number of characteristics associated with the flowstreams.

FIGURE II-2-3.1. CONTENT OF /CNTVAR/ COMMON BLOCK

2.4 The /LOCALV/ Common Block

As discussed in Subpart 1.2.1.1 of Section I, unfortunately the scratch storage used by each subroutine in FORTRAN programs is allocated core storage independently of the scratch storage allocated to other subroutines. This causes large programs such as CPSM4 with many subroutines to be very core intensive. This was a particularly severe problem in the original version of the program.

To deal with this problem, the common block /LOCALV/ is introduced. Subroutines allocate their local variables to this block, thus, reducing the total amount of scratch storage used to the maximum amount needed by a single routine. Obviously, some caution must be exercised in using this technique in cases where one routine calls another. In such cases, both must not use the same words in the block.

Operationally, many routines will contain this block; however, all variables in the block will be treated as local variables in the documentation.

3.0 OVERALL STRUCTURE

The preceding two chapters presented the overall programming and storage allocation approaches taken in the current version of program CPSM4. At this point, the overall program structure can be presented. This chapter discusses the actual organization of the subroutines, functions, and common blocks. It then gives the documentation for the main control routines.

The subroutines, functions, and common blocks in program CPSM4 are divided into six general categories. These are:

(1) The main control elements
(2) The computational utilities
(3) The unit operations
(4) Character manipulation
(5) Input processing
(6) Report generation.

This chapter presents the main control routines, while the remaining chapters present the other routines. Figure II-3-0.1 shows all subroutines, functions, and common blocks in the program as they are grouped into the above categories. It also gives a brief description of each.

In so far as the main control elements are concerned the common blocks have already been described in the previous chapter.

I Main Control Elements

Element	Description
Block Data	Initialize common blocks
CPSM4	Main control program
SIMUL	Simulation executive routine
//	Working storage
/CNTVAR/	Program control variables
/LOCALV/	Local scratch storage
/WORK/	Control working storage utilization

II The Computational Utilities

Element	Description
CMPBRK	Compute a breakage matrix.
BRSIZE	Compute bridge based on size intervals
INTS	Perform linear interpolation
YINTRP	Perform 4-point Lagrangian interpolation
SDSIZE	Summarize flowstream data by size increment
DSTRB	Split feed according to a distribution curve
ECON	Compute economic data

III The Unit Operations

Element	Description
BREAK2	Simulate breakage that occurs during operation of a unit
NOTPTH	Compute proportion not passing through hole
SCREEN	Simulate screening unit operation
DCWASH	Compute washing distribution curve
WASH	Simulate washing unit operation
WASH2	Simulate two-stage washing unit operation
FROTH	Simulate froth flotation unit operation
CRUSH	Simulate crushing unit operation
BREAK	Simulate breaking process
ROTARY	Simulate rotary breaking unit operation

IV Character Manipulation

Element	Description
/CHRSET/	Define character set
/CODERC/	Contain raw coded record
/FILES/	Define file names
PRSIZE	Print size values
RDDESC	Read descriptor
PRDESC	Print descriptor
CNVTIE	Convert value to display form
/CONST/	Store constant values
RDVAL	Convert value to internal numeric form

FIGURE II-3-0.1. ELEMENTS OF PROGRAM

V Input Processing

Element	Description
RDUNIT	Read unit information
RDCURV	Read equipment type information
RDFEED	Read feed description
CONFSV	Create a configuration data base
FEEDSV	Create a feed data base
RDBASE	Read a data base
RDCFBS	Read the configuration from a data base
RDFDBS	Read the feed description from a data base
SVBASE	Create a data base

IV Report Generation

Element	Description
VESSEL	Print unit description
PRHEAD	Print report heading
CMPUS	Compute unit summary values
/BLK1/	Store unit summary values
/BLK2/	Sort distribution curve values
OUTPT1	Output flowstream specific gravity analysis
OUTPT2	Output unit summary data
OUTPT4	Output summaries for units and flowstreams
PRLINE	Print a line of output
/IOCNT/	Store carriage control values
/FLAGS/	Store line width for output

FIGURE II-3-0.1. (continued)

3.1 The BLOCK DATA Subroutine

Within most versions of FORTRAN all variables in labeled common blocks must be initialized in a BLOCK DATA subroutine. The BLOCK DATA subroutine shown in Figure II-3-1.1 initializes the common blocks for program CPSM4. The various parts of this chapter discussing the individual common blocks explain the significance of the individual initializations.

```
 BLOCK DATA
 COMMON/CHRSET/LETTER(47)
 COMMON/CNTVAR/KF(2,100),SIZE(23),GBOUND(21),GRAV(20),WFLOW(100)
1             ,S(100),D1(60),D2(60),D3(60),L(60)
2                              ,NSIZE,NGRAV,IOUT,NUNITS,NFLOWS,NCHAR
 COMMON/CONST/TENS(9),ICOM
 COMMON/FILES/READER,PRINTR,OFFLIN
 COMMON/FLAGS/LINEW
 COMMON/WORK/NEXT,LAST,ICURVE,IUNIT,IDATA,ITITLE,ICHAR,IFLW,IASH,
1 ISUL,IBTU
 INTEGER READER,PRINTR,OFFLIN
 DATA READER/5/
 DATA PRINTR/6/
 DATA OFFLIN/0/
 DATA NEXT/1/
 DATA LAST/7000/
 DATA NFLOWS/1/
 DATA KF(1,1),KF(2,1)/0,0/
 DATA WFLOW(1)/0.0/
 DATA S(1)/1HC/
 DATA LINEW/135/
 DATA ICOM/3/
 DATA TENS(1),TENS(2),TENS(3),TENS(4)/10.,100.,1000.,10000./
 DATA TENS(5),TENS(6),TENS(7)/100000.,1000000.,10000000./
 DATA TENS(8),TENS(9)/100000000.,1000000000./
 DATA LETTER(1),LETTER(2),LETTER(3),LETTER(4)/1HA,1HB,1HC,1HD/
 DATA LETTER(5),LETTER(6),LETTER(7),LETTER(8)/1HE,1HF,1HG,1HH/
 DATA LETTER(9),LETTER(10),LETTER(11),LETTER(12)/1HI,1HJ,1HK,1HL/
 DATA LETTER(13),LETTER(14),LETTER(15),LETTER(16)/1HM,1HN,1HO,1HP/
 DATA LETTER(17),LETTER(18),LETTER(19),LETTER(20)/1HQ,1HR,1HS,1HT/
 DATA LETTER(21),LETTER(22),LETTER(23),LETTER(24)/1HU,1HV,1HW,1HX/
 DATA LETTER(25),LETTER(26),LETTER(27),LETTER(28)/1HY,1HZ,1H0,1H1/
 DATA LETTER(29),LETTER(30),LETTER(31),LETTER(32)/1H2,1H3,1H4,1H5/
 DATA LETTER(33),LETTER(34),LETTER(35),LETTER(36)/1H6,1H7,1H8,1H9/
 DATA LETTER(37),LETTER(38),LETTER(39),LETTER(40)/1H+,1H-,1H*,1H//
 DATA LETTER(41),LETTER(42),LETTER(43),LETTER(44)/1H(,1H),1H$,1H=/
 DATA LETTER(45),LETTER(46),LETTER(47)/1H ,1H,,1H./
 END
```

FIGURE II-3-1.1. LISTING OF BLOCK DATA SUBROUTINE

3.2 Program CPSM4

Program CPSM4 is the main control routine. It directs the processing of the input, the execution of the simulation, and the generation of the summary output tables. The particular procedure followed is shown in Figure II-3-2.1,which contains the listing and basic documentation for program CPSM4.

The reader should be aware that the PROGRAM statement itself is unique to CDC-FORTRAN and should be removed prior to compiling it on non-CDC machines.

```
      PROGRAM CPSM4(INPUT,OUTPUT    ,TAPE5=INPUT,TAPE6=OUTPUT,TAPE3,
     1 TAPE4)
C
C     TITLE(COAL PREPARATION SIMULATION MODEL, VERSION 4)
C
C     PROGRAM CPSM4 IS THE MAIN CONTROL ROUTINE FOR THE COAL PREPAR-
C     ATION MODEL SIMULATION PROGRAM.
C
C     EXTERNALS    CNVTIE, ECON, OUTPT1, OUTPT4, PRDESC, PRHEAD,
C                  PRLINE, RDBASE, RDCFBS, RDDESC, RDFDBS, SIMUL
C                  CONFSV, FEEDSV
C
C     COMMON       VARIABLES USED
C     //           SG
C     /CODERC/     ICOL, LCOL, LINE
C     /FILES/      READER
C     /CNTVAR/     GBOUND, IOUT, KF, NCHAR, NFLOWS, NGRAV, NSIZE,
C                  S,SIZE.
C     /IOCNT/      CARCON, JOB
C     /WORK/       ICURVE, IDATA, ITITLE, IUNIT, NEXT
C     /CHRSET/     LETTER
C     /LOCALV/     LTITLE
C
C     VARIABLE     DESCRIPTION
C     FILE3        THIS INTEGER ARRAY IS USED TO CONTAIN DETAILS OF THE
C                  CONFIGURATION DATA BASE.
C     I            THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                  MOVEMENT THROUGH LOCAL LOOPS.
C     IF           THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                  MOVEMENT THROUGH THE FLOWS.
C     LD1,LD2,LD3, THESE INTEGER ARRAYS CONTAIN INTERNAL DISPLAY CODE
C     LD4,LD5,LD6, VALUES FOR OUTPUT PURPOSES.
C     LD7
C
C     NCELL        THIS DUMMY INTEGER VARIABLE IS USED TO CONTAIN THE
C                  NUMBER OF CELLS OF WORKING STORAGE REQUIRED BY VAR-
C                  IOUS ENTITIES.
C     NCOST        THIS INTEGER VARIABLE IS USED TO CONTROL EXECUTION OF
C                  SUBROUTINE  E C O N.
C     NFEED        THIS INTEGER VARIABLE CONTAINS THE NUMBER OF INPUT
C                  FEEDS
C     NOW          THIS INTEGER VARIABLE CONTAINS THE POSITION IN
C                  WORKING STORAGE OF THE FEED DESCRIPTION.
C     LINES        THIS INTEGER VARIABLE IS USED TO KEEP TRACK OF THE
C                  NUMBER OF LINES USED ON THE CURRENT PAGE OF THE RE-
C                  PORTS PRODUCED BY THE ROUTINE.
C     IBASE        THIS INTEGER VARIABLE CONTROLS INPUT FROM DATABASES.
C     TFLOW        THIS REAL VARIABLE CONTAINS THE FLOWRATE OF THE
C                  INPUT FLOWSTREAM.
```

FIGURE II-3-2.1. LISTING OF PROGRAM CPSM4

```
C     NREC           THIS INTEGER VARIABLE IS USED TO CONTAIN THE LENGTH
C                    OF THE CONFIGURATION HEADER RECORD.
C
      COMMON/LOCALV/LVFILL(135),LTITLE(65)
      COMMON/CODERC/ICOL,LCOL,LINE(150)
      COMMON/FILES/READER,PRINTR,OFFLIN
      COMMON/CNTVAR/KF(2,100),SIZE(23),GBOUND(21),GRAV(20)
     1,WFLOW(100) ,S(100),D1(60),D2(60),D3(60),L(60)
     2                          ,NSIZE,NGRAV,IOUT,NUNITS,NFLOWS,NCHAR
      COMMON/WORK/NEXT,LAST,ICURVE,IUNIT,IDATA,ITITLE,ICHAR,IFLW,IASH,
     1 ISUL,IBTU
      COMMON/CHRSET/LETTER(47)
      COMMON/IOCNT/JOB,CARCON
      COMMON SG(7000)
      DIMENSION FILE3(63)
      DIMENSION LD1(5),LD2(5),LD3(6),LD4(5),LD5(5),LD6(5),LD7(3)
      INTEGER FILE3,SG
      INTEGER CARCON
      INTEGER READER,PRINTR,OFFLIN,S
      DATA LD1(1),LD1(2),LD1(3),LD1(4)/17,103871955,340283469,758731586/
      DATA LD1(5)/338/
      DATA LD2(1),LD2(2),LD2(3),LD2(4)/20,256414153,246836053,237325134/
      DATA LD2(5)/355737938/
      DATA LD3(1),LD3(2),LD3(3),LD3(4)/25,68498697,235225679,246836053/
      DATA LD3(5),LD3(6)/237325134,355737938/
      DATA LD4(1),LD4(2),LD4(3),LD4(4)/20,588701761,246690753,213189775/
      DATA LD4(5)/72627498/
      DATA LD5(1),LD5(2),LD5(3),LD5(4)/19,691310852,203743699,759243716/
      DATA LD5(5)/5518634/
      DATA LD6(1),LD6(2),LD6(3),LD6(4)/16,692605333,320263186,252793044/
      DATA LD6(5)/42/
      DATA LD7(1),LD7(2),LD7(3)/6,689459524,42/
C
C     STEP 1         INITIALIZE VARIABLES FOR THE RUN AND READ THE MAIN
C                    CONTROL CARD
C
    4 FILE3(1)=3
      FILE3(3)=0
      NEXT=1
      READ(READER,100) (LTITLE(I),I=1,65),IBASE,NCOST,IOUT
  100 FORMAT(65A1,3I5)
C
C     STEP 2         IF THIS IS A COST RUN ONLY BRANCH TO STEP 4.
C
      IF(IOUT.LT.0) GOTO 11
C
C     STEP 3         OPEN THE CONFIGURATION DATA BASE FILE.
C
      IF(IBASE.EQ.1) GOTO 12
```

FIGURE II-3-2.1. (Continued)

```
      CALL CONFSV
      CALL FEEDSV
   12 CONTINUE
      CALL RDBASE(FILE3,SG,NREC,0)
      NEXT=NREC
      IUNIT=NEXT
      ICURVE=SG(NEXT)
C
C     STEP 4        USING SUBROUTINE  R D D E S C PROCESS AND STORE
C                   THE RUN DESCRIPTION.
C
   11 ICOL=1
      LCOL=65
      DO 13 I=ICOL,LCOL
   13 LINE(I)=LTITLE(I)
      CALL RDDESC(SG(NEXT),NCELL)
      ITITLE=NEXT
      NEXT=NEXT+NCELL
      DO 10 I=1,150
   10 LINE(I)=LETTER(45)
C
C     STEP 5        IF THIS IS A COST RUN ONLY BRANCH TO STEP 13.
C
      IF(IOUT.LT.0) GOTO 101
C
C     STEP 6        WITH THE HELP OF SUBROUTINE  R D C F B S READ AND
C                   STORE THE CONFIGURATION DESCRIPTION
C
      CALL RDCFBS(FILE3)
C
C     STEP 7        USING  P R D E S C,  C N V T I E  AND  P R H E A D
C                   PRINT THE FLOWSTREAM DEFINITION.
C
      LINES=52
      DO 6 IF=1,NFLOWS
      IF(LINES.LT.51)GO TO 25
      LINES=0
      CALL PRHEAD
      JOB=-1
      CARCON=LETTER(45)
      ICOL=7
      CALL PRDESC(LD1)
      ICOL=ICOL+15
      CALL PRDESC(LD2)
      ICOL=ICOL+16
      CALL PRDESC(LD3)
      CALL PRLINE
   25 LINES=LINES+2
      CALL PRLINE
```

FIGURE II-3-2.1. (Continued)

```
      ICOL=14
      CALL CNVTIE(IF,2,-1)
      ICOL=ICOL+32
      CALL CNVTIE(KF(1,IF),2,-1)
      ICOL=ICOL+4
      LINE(ICOL)=S(IF)
      ICOL=ICOL+30
      CALL CNVTIE(KF(2,IF),2,-1)
      CALL PRLINE
      IF(KF(1,IF).NE.0) GOTO 5
      CARCON=LETTER(37)
      ICOL=18
      CALL PRDESC(LD7)
      CALL PRLINE
    5 CONTINUE
      IF(KF(2,IF).NE.0)GO TO 6
      IF(S(IF).NE.LETTER(3)) GOTO 7
      CARCON=LETTER(37)
      ICOL=18
      CALL PRDESC(LD4)
      CALL PRLINE
    7 CONTINUE
      IF(S(IF).NE.LETTER(13)) GOTO 8
      CARCON=LETTER(37)
      ICOL=18
      CALL PRDESC(LD5)
      CALL PRLINE
    8 CONTINUE
      IF(S(IF).NE.LETTER(18)) GOTO 6
      CARCON=LETTER(37)
      ICOL=18
      CALL PRDESC(LD6)
      CALL PRLINE
    6 CARCON=LETTER(45)
      NOW=NEXT
C
C     STEP 8        WITH THE HELP OF SUBROUTINE  R D F O B S  READ AND
C                   STORE THE DESCRIPTION OF THE FEED TO THE CONFIGUR-
C                   ATION.
C
   70 CONTINUE
      NEXT=NOW
      CALL RDFOBS(NFEED,TFLOW)
C
C     STEP 9        IF THE NUMBER OF FEEDS IS LESS THAN ZERO A NEW
C                   CONFIGURATION IS TO BE READ IN SO BRANCH TO STEP 1.
C                   IF THE NUMBER OF FEEDS IS ZERO BRANCH TO STEP 15.
C
      IF(NFEED) 4,999,9
```

FIGURE II-3-2.1. (Continued)

```
C
C     STEP 10        CALL SUBROUTINE  O U T P T 1 TO DISPLAY
C                    THE FEED FLOWSTREAM.
C
    9 CONTINUE
      CALL OUTPT1(SG(IDATA),SIZE,GBOUND,NSIZE,NGRAV,NCHAR,TFLOW,1,KF,S)
C
C     STEP 11        USING SUBROUTINE  S I M U L  SIMULATE THE PASSAGE OF
C                    OF THE FEED THROUGH THE CONFIGURATION.
C
   60 CALL SIMUL
C
C     STEP 12        USING SUBROUTINE  O U T P T 4  PRODUCE THE UNIT AND
C                    FLOWSTREAM SUMMARY REPORTS.
C
      CALL OUTPT4
C
C     STEP 13        IF THE COST COMPONENT IS TO BE RUN CALL  E C O N.
C
  101 IF(NCOST.NE.0) CALL ECON
C
C     STEP 14        BRANCH TO STEP 8 TO READ IN THE NEW FEEDS.
C
      GOTO 70
C
C     STEP 15        END EXECUTION OF PROGRAM.
C
  999 STOP
      END
```

FIGURE II-3-2.1. (Continued)

3.3 Subroutine SIMUL

Subroutine SIMUL is used to physically direct the simulation of the passage of the feed flowstream through the configuration. Figure II-3-3.1 contains the listing and basic documentation for subroutine SIMUL.

```
      SUBROUTINE SIMUL
C
C     TITLE(SIMULATE PERFORMANCE OF CONFIGURATION)
C
C
C     SUBROUTINE SIMUL IS THE MAIN ROUTINE FOR CONTROLLING THE SIMULA-
C     TION OF THE CONFIGURATION VERSUS THE FEED DESCRIPTION.
C
C     EXTERNALS    BREAK2,CMPUS,CNVTIE,CRUSH,FROTH,OUTPT1,OUTPT2,PRDESC,
C                  PRLINE,ROTARY,SCREEN,SDSIZE,WASH,WASH2
C
C     COMMON       VARIABLES USED
C     /CHRSET/     LETTER
C     /WORK/       NEXT,LAST,IFLW
C     /CNTVAR/     KF,SIZE,GROUND,GRAV,S,D1,D2,D3,L,NSIZE,NGRAV,IOUT,
C                  NUNITS,NFLOWS,NCHAR,WFLOW
C     /BLK1/       ISCC,ISFO,ISM,ISMX,ISRF,SUMRY
C     /IOCNT/      CARCON
C     /CODERC/     ICOL,LINE
C     /BLK2/       DISTRB
C     //           WS
C
C     VARIABLE     DESCRIPTION
C     IFLOW        THIS 1-DIMENSIONAL INTEGER ARRAY IS USED TO CONTAIN
C                  THE ADDRESSES IN WORKING STORAGE OF THE BEGINNINGS
C                  OF THE DETAILED FLOW STORAGE ARRAYS.
C      IPRAM,IP1-  THESE INTEGER VARIABLES ARE USED TO CONTAIN THE
C       IP4        FLOWSTREAM NUMBERS OF THE FLOWS ASSOCIATED WITH THE
C                  CURRENT UNIT.
C      IADR,I1-I7  THESE INTEGER VARIABLES ARE USED TO CONTAIN THE
C                  ADDRESSES IN WORKING STORAGE OF THE STORAGE AREAS
C                  ASSOCIATED WITH THE CURRENT UNIT.
C      IOST        THIS INTEGER 1-DIMENSIONAL ARRAY IS USED TO CONTAIN
C                  THE OUTPUT FLOW TYPE SYMBOLS ASSOCIATED WITH THE
C                  CURRENT UNIT.
C      NS          THIS INTEGER VARIABLE CONTAINS THE ROW NUMBER IN
C                  THE SUMMARY ARRAYS OF THE VALUES RELATING TO THE
C                  COMPOSITE.
C      NCELL       THIS INTEGER VARIABLE CONTAINS THE TOTAL NUMBER
C                  OF CELLS REQUIRED TO STORE EACH SET OF DETAILED
C                  FLOWSTREAM DATA.
C      IAVL,AVL    THIS TYPELESS VARIABLE IS THE STORAGE AREA AVAILAB-
C                  ILITY POINTER. IF IT IS ZERO THERE ARE NO AREAS
C                  AVAILABLE. IF IT IS POSITIVE IT CONTAINS THE ADDRESS
C                  OF THE FIRST AVAILABLE AREA. AT THAT ADDRESS IS
C                  STORED THE ADDRESS OF THE NEXT AVAILABLE AREA OR A
C                  ZERO. THIS CONTINUES TO CHAIN ALL AREAS AVAILABLE.
C      IU          THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                  MOVEMENT THROUGH THE UNITS IN THE CONFIGURATION.
C      NP          THIS INTEGER VARIABLE CONTAINS THE NUMBER OF FLOW-
C                  STREAM PARAMETERS ASSOCIATED WITH THE CURRENT UNIT.
C      IEQUIP      THIS INTEGER VARIABLE CONTAINS THE ADDRESS IN
C                  WORKING STORAGE OF THE EQUIPMENT TABLE ASSOCIATED
C                  WITH THE CURRENT UNIT.
C      IUO,UO      THIS TYPELESS VARIABLE CONTAINS THE UNIT OPER-
C                  ATION CODE.
```

FIGURE II-3-3.1. LISTING OF SUBROUTINE SIMUL

```
C     NIN            THIS INTEGER VARIABLE CONTAINS THE NUMBER OF UNIT
C                    INPUT FLOWS.
C     FLOWM          THIS REAL VARIABLE CONTAINS THE TOTAL AMOUNT OF
C                    MATERIAL BEING INPUT INTO THE UNIT.
C     F              THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                    MOVEMENT THROUGH THE FLOWS.
C     I,J            THESE DUMMY INTEGER VARIABLES ARE USED TO CONTROL THE
C                    MOVEMENT THROUGH MISCELLANEOUS LOCAL LOOPS.
C     IA,IB,IC,ID    THESE DUMMY INTEGER VARIABLES ARE USED TO STORE FLOW
C                    INFORMATION ADDRESSES IN WORKING STORAGE.
C     NINT           THIS INTEGER VARIABLE CONTAINS THE NUMBER OF UNIT
C                    INTERMEDIATE SCRATCH STORAGE AREAS.
C     NOUT           THIS INTEGER VARIABLE CONTAINS THE NUMBER OF UNIT
C                    OUTPUT FLOWS.
C     IO           . THIS INTEGER VARIABLE CONTAINS THE STARTING ADDRESS
C                    IN THE UNIT PARAMETER ARRAYS OF THE OUTPUT FLOW
C                    PARAMETERS.
C     LO             THIS INTEGER VARIABLE CONTAINS THE ENDING ADDRESS
C                    IN THE UNIT PARAMETER ARRAYS OF THE OUTPUT FLOW
C                    PARAMETERS.
C     K              THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                    MOVEMENT THROUGH THE FLOW CHARACTERISTICS.
C     NBRK,XBRK      THIS TYPELESS VARIABLE IS USED TO CONTAIN THE
C                    NUMBER OF CELLS USED IN THE SPECIFICATION OF THE
C                    BREAKAGE UNIT.
C     IBR,BR         THIS TYPELESS VARIABLE SPECIFIES WHETHER BREAKAGE
C                    TAKES PLACE DURING OPERATION OF THE UNIT.
C     DV             THIS REAL VARIABLE IS USED TO CONTAIN THE VALUE OF
C                    DECISION VARIABLES FOR THE MOISTURE CONTENT OF OUTPUT
C                    FLOWSTREAMS.
C     FIN,F1         THESE INTEGER VARIABLES ARE USED AS ARRAY POINTERS.
C     FL             THIS REAL VARIABLE CONTAINS THE PROPORTION OF
C                    MATERIAL IN THE LOWER FLOWSTREAM.
C     FU             THIS REAL VARIABLE CONTAINS THE PROPORTION OF
C                    MATERIAL IN THE UPPER FLOWSTREAM.
C     IAD            THIS INTEGER ARRAY IS EQUIVALENCED TO IA, IB, IC, ID.
C     IBLK1          THIS INTEGER VARIABLE CONTAINS FOUR TIMES THE
C                    MAXIMUM NUMBER OF CHARACTERISTICS.
C     ICONV          THIS INTEGER VARIABLE IS USED AS AN INDICATOR OF
C                    CONVERGENCE IF RETREATMENT STREAMS ARE PRESENT.
C     IPASS          THIS INTEGER VARIABLE COUNTS THE NUMBER OF
C                    ITERATIONS.
C     IRTRT          THIS ONE-DIMENSIONAL INTEGER ARRAY CONTAINS THE
C                    SEQUENCE NUMBERS OF THE RETREATMENT STREAMS.
C     JF,JF1,JFIN,   THESE INTEGER VARIABLES ARE USED AS DUMMY ARRAY
C     JJ,JP2,JP3,    POINTERS.
C     JP4,KK
C
C     LD1-LD7        THESE INTEGER ARRAYS CONTAIN CHARACTER CODES FOR
C                    DISPLAY.
C     NDATA          THIS INTEGER VARIABLE CONTAINS THE NUMBER OF CELLS
C                    REQUIRED TO STORE THE DETAILED FLOW VALUES.
C     NR             THIS INTEGER VARIABLE CONTAINS THE NUMBER OF RETREAT-
C                    MENT STREAMS.
C     OLDF,OLDF1     THESE REAL ARRAYS CONTAIN DETAILED FLOWSTREAM
C                    VALUES FROM PREVIOUS ITERATIONS.
```

FIGURE II-3-3.1. (Continued)

Reproduced from best available copy.

```
C     OLDWF         THIS REAL ARRAY CONTAINS MOISTURE CONTENT VALUES
C                   FROM PREVIOUS ITERATIONS.
C     REM           THIS REAL VARIABLE IS USED FOR TEMPORARY STORAGE
C                   FOR THE WATER CONTENT.
C     RU,RL         THESE REAL VARIABLES ARE USED FOR THE UPPER AND
C                   LOWER FLOWSTREAM CONTENT RATIOS FOR SPLITTERS.
C     SUM           THIS REAL VARIABLE CONTAINS THE TOTAL WATER CONTENT
C                   OF THE STREAMS ENTERING A BLENDER.
C     TEMP          THIS REAL VARIABLE IS USED FOR TEMPORARY STORAGE.
C     TOTAL         THIS REAL VARIABLE CONTAINS THE RATE OF FLOW OF A
C                   FLOWSTREAM.
C
      COMMON/CHRSET/LETTER(47)
      COMMON/WORK/NEXT,LAST,ICURVE,IUNIT,IDATA,ITITLE,ICHAR,IFLW
      COMMON/CNTVAR/KF(2,100),SIZE(23),GBOUND(21),GRAV(20)
     1,WFLOW(100) ,S(100),D1(60),D2(60),D3(60),L(60)
     2                         ,NSIZE,NGRAV,IOUT,NUNITS,NFLOWS,NCHAR
      COMMON/BLK1/ISFD,ISCC,ISRF,ISMX,SYLD(24),TYLD(24),EFFIC(24),
     1 BTUREC(24),ASHEPR(24),SFLT(24),SINK(24),SMISPL(24),SNRGR(24),
     2 SGRAV(24),SPE(24),SIMP(24),SEA(24),SUMRY(24,40)
      COMMON/CODERC/ICOL,LCOL,LINE(150)
      COMMON/BLK2/DISTRB(24,20)
      COMMON WS(6000)
      COMMON/IOCNT/JOB,CARCON
      DIMENSION LD1(9),LD2(8),LD3(7),LD4(5),LD5(5),LD6(10)
      DIMENSION LD7(10)
      DIMENSION IFLOW(100),IPRAM(7),IADR(7),IOST(3)
      DIMENSION IAD(4)
      DIMENSION IRTRT(10),OLDF(10,5),OLDWF(10)
      DIMENSION OLDF1(10,2)
      INTEGER S,F,FIN,F1
      INTEGER CARCON
      DIMENSION ISM(4)
      EQUIVALENCE (ISFD,ISM(1))
      EQUIVALENCE (UO,IUO),(BR,IBR),(IAVL,AVL),(XBRK,NBRK)
      EQUIVALENCE (IA,IAD(1)),(IB,IAD(2)),(IC,IAD(3)),(ID,IAD(4))
      EQUIVALENCE (IADR(1),I1),(IADR(2),I2),(IADR(3),I3),(IADR(4),I4)
     1            ,(IADR(5),I5),(IADR(6),I6),(IADR(7),I7)
      EQUIVALENCE (IPRAM(1),IP1),(IPRAM(2),IP2),(IPRAM(3),IP3)
     1            ,(IPRAM(4),IP4)
      DATA LD1(1),LD1(2),LD1(3),LD1(4)/38,339540813,20552515,134553667/
      DATA LD1(5),LD1(6),LD1(7)/336929363,337917165,104672066/
      DATA LD1(8),LD1(9)/204327763,157831168/
      DATA LD2(1),LD2(2),LD2(3),LD2(4)/31,134560589,256400212,134540115/
      DATA LD2(5),LD2(6),LD2(7)/157471310,274025299,340283469/
      DATA LD2(8)/19/
      DATA LD3(1),LD3(2),LD3(3),LD3(4)/27,220542153,236900942,274025286/
      DATA LD3(5),LD3(6),LD3(7)/205355846,256562510,596/
      DATA LD4(1),LD4(2),LD4(3),LD4(4)/16,257246229,347366159,397755984/
      DATA LD4(5)/5/
      DATA LD5(1),LD5(2),LD5(3),LD5(4)/17,758420691,154696518,256562510/
      DATA LD5(5)/596/
      DATA LD6(1),LD6(2),LD6(3),LD6(4)/43,154744134,107035461,240309711/
      DATA LD6(5),LD6(6),LD6(7)/304911239,760038354,186355587/
      DATA LD6(8),LD6(9),LD6(10)/87082221,104672085,57940/
      DATA LD7(1),LD7(2),LD7(3),LD7(4)/42,664697294,263467982,370483653/
```

FIGURE II-3-3.1. (Continued)

```
      DATA LD7(5),LD7(6),LD7(7)/236739393,105927853,477024852/
      DATA LD7(8),LD7(9),LD7(10)/88610057,255408615,2535/
      DATA IBLK1/40/
C
C     STEP 1        COMPUTE THE ROW NUMBER IN THE SUMMARY ARRAYS OF THE
C                   VALUES FOR THE COMPOSITE SIZE FRACTIONS.
C
      NS=NSIZE+1
      ISCC=1
      ISRF=NCHAR+1
      ISMX=2*NCHAR+1
      ISFD=3*NCHAR+1
      IF(4*NCHAR.LE.IBLK1) GOTO 4
      CARCON=LETTER(27)
      ICOL=0
      CALL PRDESC(LD1)
      CALL PRLINE
      STOP
    4 CONTINUE
C
C     STEP 2        COMPUTE THE NUMBER OF CELLS REQUIRED TO STORE THE
C                   DETAILED FLOW VALUES.
C
      NCELL=NSIZE*NGRAV*NCHAR
      NDATA=NCELL
      IF(NSIZE*NSIZE.GT.NCELL) NCELL=NSIZE*NSIZE
C
C     STEP 3        SET THE FLOW VALUE STORAGE AVAILABILITY INDICATOR
C                   TO INDICATE THAT NO STORAGE AREAS ARE AVAILABLE.
C
      IAVL=0
C
C     STEP 4        ALLOCATE THE STORAGE AREA BEING OCCUPIED BY THE
C                   DETAILED FEED DESCRIPTION DATA, AND USING SUBROUTINE
C                   S D S I Z E  COMPUTE THE SUMMARY DATA FOR THE FEED.
C
      IFLOW(1)=NEXT
      CALL SDSIZE(WS(NEXT),SUMRY(1,ISFD),NSIZE,NGRAV,NCHAR,TOTAL)
      WS(IFLW+1)=TOTAL
      DO 5 K=2,NCHAR
      JJ=ISM(1)-1+K
      JF=IFLW+K
      WS(JF)=SUMRY(NS,JJ)
    5 CONTINUE
      NEXT=NEXT+NCELL
      NR=0
      IPASS=0
C
C     STEP 5        INITIALIZE AT ONE THE DUMMY COUNTER TO CONTROL THE
C                   MOVEMENT THROUGH THE UNITS IN THE CONFIGURATION.
C
    6 CONTINUE
      IPASS=IPASS+1
      IF(IPASS.EQ.11) NR=0
      IF(IPASS.GT.11) GOTO 184
      DO 185 IU=1,NUNITS
```

FIGURE II-3-3.1. (Continued)

```
C
C     STEP 5        INITIALIZE AT ZERO THE TOTAL NUMBER OF DETAILED
C                   FLOW AREA PARAMETERS NEEDED BY THE CURRENT UNIT
C                   OPERATION.
C
      NP=0
C
C     STEP 7        OBTAIN THE BASE ADDRESS OF THE EQUIPMENT DESCRIPTION
C                   TABLE FOR THE CURRENT UNIT.
C
      IEQUIP=L(IU)
C
C     STEP 8        OBTAIN THE UNIT OPERATION TYPE CODE FROM THE EQUIP-
C                   MENT TABLE.
C
      UO=WS(IEQUIP+2)
      BR=WS(IEQUIP+4)
C
C     STEP 9        DETERMINE THE NUMBER OF INPUT FLOWS TO THE UNIT.
C
      NIN=1
C
C     STEP 10       INITIALIZE THE TOTAL AMOUNT OF MATERIAL IN THE
C                   INPUT FLOWS AT ZERO.
C
      FLOWM=0.0
C
C     STEP 11       INITIALIZE AT ZERO THE DUMMY COUNTER USED TO CONTROL
C                   THE MOVEMENT THROUGH THE FLOWS.
C
      F=0
C
C     STEP 12       INITIALIZE THE NUMBER OF INTERMEDIATE FLOWS REQUIRED
C                   BY THE UNIT AT ZERO.
C
      NINT=0
      IF(IUO.EQ.1) GOTO 25
C
C     STEP 13       INITIALIZE AT ONE THE DUMMY COUNTER USED TO MOVE
C                   THROUGH THE INPUT FLOWS TO THE UNIT.
C
      DO 15 I=1,NIN
C
C     STEP 14       INCREMENT BY ONE THE DUMMY COUNTER USED TO CONTROL
C                   THE MOVEMENT THROUGH THE FLOWS, AND IF IT EXCEEDS
C                   THE NUMBER OF FLOWS BRANCH TO STEP 59.
C
   10 F=F+1
      IF(F.GT.NFLOWS)GO TO 995
C
C     STEP 15       IF THE DESTINATION OF THE CURRENT FLOW IS NOT THE
C                   CURRENT UNIT, LOOP BACK TO STEP 14.
C
      IF(KF(2,F).NE.IU)GO TO 10
C
C     STEP 16       INCREMENT THE NUMBER OF UNIT FLOW PARAMETERS BY ONE.
```

FIGURE II-3-3.1. (Continued)

```
C
      NP=NP+1
C
C     STEP 17         STORE THE FLOW NUMBER AND FLOW ADDRESS IN THE PARA-
C                     METER STORAGE ARRAYS.
C
      IPRAM(NP)=F
      IADR(NP)=IFLOW(F)
C
C     STEP 18         USING SUBROUTINE  S D S I Z E  COMPUTE THE SUMMARY
C                     VALUES FOR THE FLOW.
C
      IA=IFLOW(F)
      IB=ISM(I)
      CALL SDSIZE(WS(IA),SUMRY(1,IB     ),NSIZE,NGRAV,NCHAR,TOTAL)
C
C     STEP 19         ADD THE AMOUNT OF MATERIAL IN THE CURRENT FLOW INTO
C                     THE TOTAL AMOUNT OF MATERIAL IN THE FEED TO THE UNIT.
C
      JF=IFLW+NCHAR *(F-1)
      FLOWM=FLOWM+WS(JF+1)
C
C     STEP 20         INCREMENT BY ONE THE DUMMY COUNTER CONTROLLING THE
C                     MOVEMENT THROUGH THE UNIT INPUT FLOWS. IF ALL HAVE
C                     NOT BEEN PROCESSED, LOOP BACK TO STEP 14.
C
   15 CONTINUE
C
C     STEP 21         BRANCH ACCORDING TO THE TYPE OF UNIT OPERATION.
C                     BRANCH     UNIT OPERATION
C                     STEP 22    CRUSHING
C                     STEP 23    BLENDING
C                     STEP 24    SPLITTING, SCREENING OR FRACTIONATION.
C                     STEP 25    TWO-STAGE WASHING
C                     STEP 26    ROTARY BREAKING
C                     STEP 27    WASHING OR FROTH FLOTATION
C
      GOTO(25,30,30,45,35,45,40,20,30),IUO
C
C     STEP 22         SET THE NUMBER OF INTERMEDIATE FLOWS TO ONE FOR THE
C                     CRUSHING UNIT OPERATION.
C
   20 NINT=1
C
C     STEP 23         SET THE NUMBER OF OUTPUT FLOWS FOR THE UNIT OPER-
C                     ATIONS OF BLENDING AND CRUSHING TO ONE AND SET THE
C                     FLOW TYPE CODE TO ZERO. THEN BRANCH TO STEP 29.
C
   25 NOUT=1
      IOST(1)=0
      GO TO 55
C
C     STEP 24         SET THE NUMBER OF OUTPUT FLOWS FOR THE UNIT TO TWO
C                     AND SET THE TYPE CODES TO U AND L FOR THE SCREENING
C                     AND SPLITTING UNIT OPERATIONS. THEN BRANCH TO STEP 29.
C
```

FIGURE II-3-3.1. (Continued)

Reproduced from best available copy.

```
   30 NOUT=2
      IOST(1)=LETTER(21)
      IOST(2)=LETTER(12)
      GO TO 55
C
C     STEP 25       SET THE NUMBER OF OUTPUT FLOWS FOR THE UNIT OPER-
C                   ATION OF TWO-STAGE WASHING TO THREE AND THE TYPE CODE
C                   FOR THE THIRD FLOW TO M. THEN BRANCH TO STEP 28.
C
   35 NOUT=3
      IOST(3)=LETTER(13)
      GO TO 50
C
C     STEP 26       SET THE NUMBER OF INTERMEDIATE FLOWS TO FOUR FOR
C                   THE UNIT OPERATION OF ROTARY BREAKING.
C
   40 NINT=4
C
C     STEP 27       SET THE NUMBER OF OUTPUT FLOWS TO TWO.
C
   45 NOUT=2
C
C     STEP 28       SET THE FLOW TYPE CODES TO C AND R.
C
   50 IOST(1)=LETTER(3)
      IOST(2)=LETTER(18)
C
C     STEP 29       INITIALIZE AT ONE THE DUMMY COUNTER TO BE USED TO
C                   CONTROL THE MOVEMENT THROUGH THE OUTPUT FLOWS FROM
C                   THE UNIT.
C
   55 DO 75 I=1,NOUT
C
C     STEP 30       INITIALIZE THE DUMMY COUNTER TO CONTROL THE MOVEMENT
C                   THROUGH THE FLOWS AT ZERO.
C
      F=0
C
C     STEP 31       INCREMENT BY ONE THE DUMMY COUNTER TO CONTROL THE
C                   MOVEMENT THROUGH THE FLOWS AND IF ALL FLOWS HAVE BEEN
C                   PROCESSED, BRANCH TO STEP 50.
C
   60 F=F+1
      IF(F.GT.NFLOWS)GO TO 996
C
C     STEP 32       IF THE CURRENT FLOW DOES NOT HAVE THE CURRENT UNIT
C                   AS AN ORIGIN, LOOP BACK TO STEP 31.
C
      IF(KF(1,F).NE.IU)GO TO 60
C
C     STEP 33       IF THE CURRENT OUTPUT FLOW CODE IS ZERO, BRANCH TO
C                   STEP 35.
C
      IF(IOST(I).EQ.0)GO TO 65
C
C     STEP 34       IF THE CURRENT OUTPUT FLOW CODE IS NOT EQUAL TO THE
```

FIGURE II-3-3.1. (Continued)

```
C                 CURRENT FLOW TYPE CODE, LOOP BACK TO STEP 31.
C
      IF(S(F).NE.IOST(I))GO TO 60
C
C     STEP 35     THE CURRENT FLOW IS AN OUTPUT FLOW FROM THE CURRENT
C                 UNIT. INCREMENT THE NUMBER OF UNIT FLOW PARAMETERS
C                 BY ONE.
C
   65 NP=NP+1
C
C     STEP 36     STORE THE FLOW NUMBER IN THE UNIT PARAMETER LIST.
C
      IPRAM(NP)=F
C
C     STEP 37     IF THERE ARE NO AVAILABLE FLOW DATA STORAGE AREAS,
C                 BRANCH TO STEP 39.
C
      IF(IAVL.EQ.0)GO TO 70
C
C     STEP 38     ALLOCATE THE FLOW TO THE CURRENT STORAGE AREA,
C                 REMOVE THE AREA FROM THE AVAILABLE SET, AND BRANCH
C                 TO STEP 40.
C
      IFLOW(F)=IAVL
      IADR(NP)=IAVL
      AVL=WS(IAVL)
      GO TO 75
C
C     STEP 39     CREATE A NEW STORAGE AREA. IF ADDITIONAL CELLS OF
C                 WORKING STORAGE ARE NOT AVAILABLE BRANCH TO STEP 61.
C
   70 IFLOW(F)=NEXT
      IADR(NP)=NEXT
      NEXT=NEXT+NCELL
      IF(NEXT.GT.LAST)GO TO 997
C
C     STEP 40     INCREMENT THE UNIT OUTPUT FLOW COUNTER BY ONE AND
C                 IF ALL HAVE NOT BEEN PROCESSED LOOP BACK TO STEP 30.
C
   75 CONTINUE
      DO 76 I=1,NP
      IAD(I)=IADR(I)
   76 CONTINUE
C
C     STEP 41     IF THE UNIT HAS ANY INTERMEDIATE FLOWS, ALLOCATE
C                 THEM FLOW STORAGE AREAS IF AVAILABLE. IF THERE IS
C                 INSUFFICIENT WORKING STORAGE, BRANCH TO STEP 61.
C
      IF(NINT.EQ.0)GO TO 90
      DO 85 I=1,NINT
      NP=NP+1
      IF(IAVL.EQ.0)GO TO 80
      IADR(NP)=IAVL
      AVL=WS(IAVL)
      GO TO 85
   80 IADR(NP)=NEXT
```

FIGURE II-3-3.1. (Continued)

```
      NEXT=NEXT+NCELL
      IF(NEXT.GT.LAST)GO TO 997
   85 CONTINUE
   90 CONTINUE
C
C     STEP 42        IF BREAKAGE DOES NOT OCCUR IN THE UNIT, BRANCH TO
C                    STEP 47
C
      IF(IBR.EQ.0) GOTO 86
C
C     STEP 43        COMPUTE THE NUMBER OF CELLS USED IN THE BREAKAGE
C                    SPECIFICATION ARRAY.
C
      XBRK=WS(IBR)
      NBRK=NBRK+NBRK+2
C
C     STEP 44        COMPUTE THE NEW FLOW TO THE UNIT AFTER BREAKAGE.
C
      CALL BREAK2(WS(I1),WS(I2),WS(I3),SIZE,NSIZE,NGRAV,NCHAR,WS(IBR),
     1 NBRK)
C
C     STEP 45        IF THE USER HAS REQUESTED ONLY SUMMARY REPORTS,
C                    BRANCH TO STEP 47
C
      IF(IOUT.EQ.0) GOTO 86
      IF(NR.GT.0) GOTO 86
      IF(IPASS.EQ.1) GOTO 86
C
C     STEP 46        USING THE SUBROUTINE  O U T P T 2  GENERATE THE
C                    PERFORMANCE RECORD OF THE BREAKAGE PART OF THE UNIT.
C
      JF=0
      CALL OUTPT2(SIZE,GBOUND,NSIZE,NGRAV,NCHAR,IU,JF,D1(IU),
     1 D2(IU),D3(IU))
   86 CONTINUE
C
C     STEP 47        USING THE APPROPRIATE SUBROUTINES SIMULATE THE
C                    OPERATION OF THE UNIT AND COMPUTE ITS PERFORMANCE
C                    MEASURES.
C
      GO TO(95,100,105,110,115,120,130,135),IUO
   95 DO 97 I=1,NDATA
      K=IA+I-1
   97 WS(K)=0.0
      NIN=0
      DO 98 F=1,NFLOWS
      IF(KF(2,F).NE.IU) GOTO 98
      IF(IPASS.GT.1) GOTO 89
      IF(KF(1,F).LT.KF(2,F)) GOTO 89
      NR=NR+1
      IPTRT(NR)=F
      GOTO 98
   89 CONTINUE
      NIN=NIN+1
      NP=NP+1
      IPRAM(NP)=F
```

FIGURE II-3-3.1. (Continued)

```
      IADR(NP)=IFLOW(F)
      IB=IFLOW(F)
      DO 99 I=1,NDATA
      K=IA+I-1
      KK=IB+I-1
   99 WS(K)=WS(K)+WS(KK)
   98 CONTINUE
      IF(NIN.GT.6) GOTO 994
      GO TO 140
  100 RU=D1(IU)/100.0
      RL=1.0-RU
      FIN=IPRAM(1)
      F=IPRAM(2)
      WFLOW(F)=WFLOW(FIN)*RU
      F=IPRAM(3)
      WFLOW(F)=WFLOW(FIN)*RL
      DO 102 I=1,NCELL
      WS(IB)=WS(IA)*RU
      WS(IC)=WS(IA)*RL
      IA=IA+1
      IB=IB+1
      IC=IC+1
  102 CONTINUE
      GO TO 140
  105 JP2=IFLW+NCHAR *(IP2-1)
      JP3=IFLW+NCHAR *(IP3-1)
      CALL SCREEN(WS(I1),WS(I2),WS(I3),WS(IEQUIP),SIZE,WS(JP2+1)
     1 ,          WS(JP3+1)    ,NSIZE,NGRAV,NCHAR,D1(IU))
      GO TO 140
  110 CONTINUE
      CALL WASH(WS(I1),WS(I2),WS(I3),WS(IEQUIP),SIZE,GRAV,DISTRB
     1           ,NSIZE,NGRAV,NCHAR,D1(IU))
      GO TO 125
  115 CALL WASH2(WS(I1),WS(I2),WS(I4),WS(I3),WS(IEQUIP),SIZE,GRAV
     1           ,DISTRB,NSIZE,NGRAV,NCHAR,D1(IU))
      JP4=IFLW+NCHAR *(IP4-1)
      CALL SDSIZE(WS(I4),SUMRY(1,ISMX),NSIZE,NGRAV,NCHAR,WS(JP4+1))
      GO TO 125
  120 CONTINUE
      CALL FROTH(WS(I1),WS(I2),WS(I3),SIZE,GRAV,GBOUND,NSIZE,NGRAV
     1           ,NCHAR,WS(IEQUIP))
  125 JP2=IFLW+NCHAR *(IP2-1)
      JP3=IFLW+NCHAR *(IP3-1)
      CALL SDSIZE(WS(I2),SUMRY(1,ISCO),NSIZE,NGRAV,NCHAR,WS(JP2+1))
      CALL SDSIZE(WS(I3),SUMRY(1,ISRF),NSIZE,NGRAV,NCHAR,WS(JP3+1))
      IF(IOUT.EQ.0)GO TO 140
      IF(NR.GT.0) GOTO 140
      IF(IPASS.EQ.1) GOTO 140
      FL=WS(JP2+1) / TOTAL
      FU=WS(JP3+1)  /TOTAL
      CALL CMPUS(WS(IEQUIP),WS(I1),WS(I2),WS(I3),NSIZE,NGRAV,GRAV
     1 ,SIZE,GBOUND,FL,FU)
      GO TO 140
  130 JP2=IFLW+NCHAR *(IP2-1)
      JP3=IFLW+NCHAR *(IP3-1)
      CALL ROTARY(WS(I1),WS(I2),WS(I3),WS(I4),WS(I5),WS(I6),WS(I7)
```

FIGURE II-3-3.1. (Continued)

```
     1              ,SIZE,WS(IEQUIP),NSIZE,NGRAV,NCHAR,WS(JP2+1)
     2              ,WS(JP3+1)   ,D1(IU),D2(IU),D3(IU))
      GO TO 140
  135 CALL CRUSH(WS(I1),WS(I2),WS(I3),SIZE,WS(IEQUIP),NSIZE,NGRAV
     1              ,NCHAR,D1(IU))
      JP2=IFLW+NCHAR *(IP2-1)
      WS(JP2+1)=FLOWM
C
C     STEP 48       COMPUTE STARTING AND ENDING ADDRESSES IN UNIT FLOW
C                   ARRAYS FOR THE OUTPUT FLOW PARAMETERS.
C
  140 IO=NIN+1
      LO=NIN+NCUT
C
C     STEP 49       IF THE UNIT IS A SPLITTER BRANCH TO STEP 51.  IF
C                   THE UNIT IS NOT A BLENDER BRANCH TO STEP 52.
C
      IF(IUO.EQ.2.OR.IUO.EQ.9) GOTO 142
      IF(IUO.GT.2)GO TO 150
C
C     STEP 50       THE UNIT IS A BLENDER.  COMBINE THE INPUT FLOWS AND
C                   COMPUTE THE SUMMARY DATA FOR THE OUTPUT FLOW.
C
      SUM=0.0
      DO 141 I=2,LO
      F=IPRAM(I)
  141 SUM=SUM+WFLOW(F)
      IO=1
      LO=1
      IA=IADR(1)
      F=IPRAM(1)
      JF=IFLW+NCHAR *(F-1)
      CALL SDSIZE(WS(IA),SUMRY(1,ISFD),NSIZE,NGRAV,NCHAR,WS(JF+1))
      WFLOW(F)=SUM
      IF(NP.NE.3) GOTO 150
      IA=IADR(2)
      F=IPRAM(2)
      JF=IFLW+NCHAR*(F-1)
      CALL SDSIZE(WS(IA),SUMRY(1,ISCD),NSIZE,NGRAV,NCHAR,WS(JF+1))
      IA=IADR(3)
      F=IPRAM(3)
      JF=IFLW+NCHAR*(F-1)
      CALL SDSIZE(WS(IA),SUMRY(1,ISRF),NSIZE,NGRAV,NCHAR,WS(JF+1))
      GOTO 150
C
C     STEP 51       THE UNIT IS A SPLITTER.  COMPUTE THE SUMMARY DATA
C                   FOR THE OUTPUT FLOWS.
C
  142 CONTINUE
      DO 145 I=IO,LO
      IA=IADR(I)
      F=IPRAM(I)
      JF=IFLW+NCHAR *(F-1)
      IB=ISM(I)
      CALL SDSIZE(WS(IA),SUMRY(1,IB     ),NSIZE,NGRAV,NCHAR,WS(JF+1))
  145 CONTINUE
```

FIGURE II-3-3.1. (Continued)

```
C
C     STEP 52       SAVE THE COMPOSITE CHARACTERISTICS OF ALL OF THE
C                   OUTPUT FLOWS FOR LATER REPORTING.
C
  150 CONTINUE
      FIN=IPRAM(NIN)
      JFIN=IFLW+NCHAR *(FIN-1)
      DO 160 I=IO,LO
      F=IPRAM(I)
      JF=IFLW+NCHAR *(F-1)
      IF(IUO.LE.2) GOTO 151
      IF(WS(JFIN+1) .LT.0.001) GOTO 152
      WFLOW(F)=WFLOW(FIN)*WS(JF+1)/WS(JFIN+1)
      GOTO 151
  152 WFLOW(F)=WFLOW(FIN)/FLOAT(NOUT)
  151 CONTINUE
      DO 155 K=2,NCHAR
      KK=ISM(I)-1+K
      JF1=JF+K
      WS(JF1)=SUMRY(NS,KK)
  155 CONTINUE
  160 CONTINUE
      IF(IUO.GT.6) GOTO 161
      IF(IUO.EQ.5) GOTO 162
      DV=D2(IU)
      IF(IUO.EQ.1.OR.IUO.EQ.6.OR.IUO.EQ.9) DV=D1(IU)
      IF(DV.LT.0.001) GOTO 161
      F=IPRAM(IO)
      JF=IFLW+NCHAR *(F-1)
      IF(IUO.EQ.1) GOTO 163
      REM=WFLOW(F)/(WFLOW(F)+WS(JF+1))*100.0
      IF(ABS(REM-DV).LT.0.001) GOTO 161
      WFLOW(F) =DV*WS(JF+1) /(100.0-DV)
      REM=WFLOW(FIN)-WFLOW(F)
      F=IPRAM(LO)
      WFLOW(F)=REM
      GOTO 161
  163 CONTINUE
      REM=DV*WS(JF+1) /(100.0-DV)
      IF(REM.GT.WFLOW(F)) WFLOW(F)=REM
      GOTO 161
  162 IF(D2(IU)+D3(IU).LT.0.001) GOTO 161
      F=IPRAM(IO)
      JF=IFLW+NCHAR *(F-1)
      WFLOW(F)=D2(IU)*WS(JF+1)/(100.0-D2(IU))
      I=LO
      F1=IPRAM(I)
      JF1=IFLW+NCHAR *(F1-1)
      WFLOW(F1 )=D3(IU)*WS(JF1+1) /(100.0-D3(IU))
      REM=WFLOW(FIN)-WFLOW(F)-WFLOW(F1)
      I=IO+1
      F=IPRAM(I)
      WFLOW(F)=REM
  161 CONTINUE
C
C     STEP 53       IF THE USER HAS REQUESTED ONLY SUMMARY REPORTS,
```

FIGURE II-3-3.1. (Continued)

```
C                     BRANCH TO STEP 56.
C
      IF(IOUT.EQ.0)GO TO 171
      IF(NR.GT.0) GOTO 171
      IF(IPASS.EQ.1) GOTO 171
      IF(IUO.EQ.1.AND.NP.NE.3) GOTO 169
C
C     STEP 54      USING SUBROUTINE  O U T P T 2  PRINT THE UNIT
C                  PERFORMANCE REPORT.
C
      CALL OUTPT2(SIZE,GBOUND,NSIZE,NGRAV,NCHAR,IU,IUO,D1(IU),D2(IU),
     1 D3(IU))
C
C     STEP 55      USING SUBROUTINE  O U T P T 1  GENERATE ANY FLOW-
C                  STREAM SPECIFIC GRAVITY REPORTS REQUESTED BY THE
C                  USER FROM THE OUTPUTS OF THE CURRENT UNIT.
C
  169 CONTINUE
      DO 170 I=IO,LO
      F=IPRAM(I)
      IF(IOUT.EQ.2)GO TO 165
      IF(KF(1,F).GT.0.AND.KF(2,F).GT.0)GO TO 170
  165 IA=IADR(I)
      JF=IFLW+NCHAR *(F-1)
      CALL OUTPT1(WS(IA),SIZE,GBOUND,NSIZE,NGRAV,NCHAR,WS(JF+1) ,F,KF,S)
  170 CONTINUE
  171 CONTINUE
C
C     STEP 56      RETURN THE DETAILED FLOW INFORMATION STORAGE AREAS
C                  FOR THOSE FLOWS NO LONGER REQUIRING THEM. THESE
C                  INCLUDE INPUT FLOWS TO THE UNIT, UNIT INTERMEDIATE
C                  FLOWS, AND THOSE FLOWS FROM THE UNIT LEAVING THE
C                  CONFIGURATION.
C
      DO 180 I=1,NP
      F=IPRAM(I)
      IF(I.LT.IO)GO TO 175
      IF(I.GT.LO.AND.IUO.EQ.1) GOTO 175
      IF(I.GT.LO)GO TO 176
      IF(KF(2,F).GT.0)GO TO 180
  175 IF(KF(1,F).EQ.0) GOTO 180
  176 IA=IADR(I)
      WS(IA)=AVL
      IAVL=IA
  180 CONTINUE
C
C     STEP 57      INCREMENT BY ONE THE DUMMY COUNTER CONTROLLING THE
C                  MOVEMENT THROUGH THE UNITS IN THE CONFIGURATION
C                  AND IF ALL HAVE NOT BEEN PROCESSED, LOOP BACK TO
C                  STEP 6. IF ALL HAVE BEEN PROCESSED, END LOCAL
C                  PROCESSING.
C
  185 CONTINUE
C
C     STEP 58      IF RETREATMENT STREAMS ARE PRESENT CHECK FOR
C                  CONVERGENCE.  IF CONVERGED END LOCAL PROCESSING, IF
```

FIGURE II-3-3.1. (Continued)

```
C                    NOT LOOP BACK TO STEP 5.
C
      IF(NR.EQ.0) GOTO 184
      ICONV=1
      IF(IPASS.EQ.1) GOTO 210
      DO 205 I=1,NR
      F=IPTRT(I)
      JF=IFLW+NCHAR *(F-1)
      IF(ABS(WFLOW(F)-OLDWF(I)).GT.0.05) ICONV=0
      DO 205 J=1,NCHAR
      JF1=JF+J
  205 IF(ABS(WS(JF1 )-OLDF(I,J)).GT.0.05) ICONV=0
      IF(ICONV.EQ.1.AND.IOUT.NE.0) GOTO 226
      IF(ICONV.EQ.1) GOTO 184
  210 CONTINUE
      DO 225 I=1,NR
      F=IPTRT(I)
      JF=IFLW+NCHAR *(F-1)
      IF(IPASS.EQ.1) GOTO 215
      IF(MOD(IPASS,2).EQ.0) GOTO 220
      IF(ABS(WFLOW(F) -2.*OLDWF(I) +OLDF1(I,2)).LT.0.0001) GOTO 214
      WFLOW(F) =OLDF1(I,2)-(OLDWF(I) -OLDF1(I,2))**2/(WFLOW(F)
     1 -2.*OLDWF(I) +OLDF1(I,2))
  214 IF(ABS(WS(JF+1) -2.*OLDF(I,1)+OLDF1(I,1)).LT.0.0001) GOTO 215
      OLDF1(I,1)=OLDF1(I,1)-(OLDF(I,1)-OLDF1(I,1))**2/(WS(JF+1)
     1 -2.*OLDF(I,1)+OLDF1(I,1))
      TEMP=OLDF1(I,1)/WS(JF+1)
      WS(JF+1) =OLDF1(I,1)
      IF(WS(JF+1) .LT.0.0001) GOTO 215
      IA=IFLOW(F)
      DO 224 J=1,NDATA
      K=IA+J-1
  224 WS(K)=WS(K)*TEMP
      CALL SDSIZE(WS(IA),SUMRY(1,ISFD),NSIZE,NGRAV,NCHAR,WS(JF+1) )
  215 OLDF1(I,2)=WFLOW(F)
      OLDF1(I,1)=WS(JF+1)
  220 CONTINUE
      DO 223 J=1,5
      JF1=JF+J
  223 OLDF(I,J)=WS(JF1 )
      OLDWF(I)=WFLOW(F)
  225 CONTINUE
      GOTO 6
  226 NR=0
      GOTO 6
  184 CONTINUE
      IF(IOUT.GT.0.AND.IPASS.EQ.1) GOTO 6
      IF(IPASS.LT.11) GOTO 999
      CARCON=LETTER(27)
      ICOL=0
      CALL PROESC(LD7)
      CALL PRLINE
      GO TO 999
  994 CARCON=LETTER(28)
      ICOL=0
      CALL CNVTIE(5562964,0,-2)
```

FIGURE II-3-3.1. (Continued)

```
      CALL CNVTIE(IU,4,-1)
      CALL PRDESC(LD2)
      CALL PRLINE
      GOTO 999
C
C     STEP 59            A UNIT IS MISSING AN INPUT FLOW. DISPLAY THE
C                        APPROPRIATE MESSAGE, AND END THE RUN.
C
  995 CARCON=LETTER(27)
      ICOL=0
      CALL PRDESC(LD3)
      CALL CNVTIE(IU,3,-1)
      CALL PRLINE
      GO TO 998
C
C     STEP 60            A UNIT IS MISSING AN OUTPUT FLOW. DISPLAY THE
C                        APPROPRIATE MESSAGE, AND END THE RUN.
C
  996 CARCON=LETTER(28)
      ICOL=0
      CALL PRDESC(LD4)
      ICOL=ICOL+2
      LINE(ICOL)=IOST(I)
      CALL PRDESC(LD5)
      CALL CNVTIE(IU,3,-1)
      CALL PRLINE
      GO TO 998
C
C     STEP 61            THERE IS INSUFFICIENT ROOM TO SIMULATE THE SPECIFIED
C                        CONFIGURATION. DISPLAY THE APPROPRIATE MESSAGE, AND
C                        END THE RUN.
C
  997 CARCON=LETTER(27)
      ICOL=0
      CALL PRDESC(LD6)
      CALL CNVTIE(IU,3,-1)
      CALL PRLINE
  998 STOP
  999 RETURN
      END
```

FIGURE II-3-3.1. (Continued)

4.0 COMPUTATIONAL UTILITIES

This chapter describes the computational utilities used within CPSM4. These utilities are called by the various unit operation and control routines within the program. Their function is to perform those general computational operations such as normalization, bridging, interpolation, etc. which are not specific to any one of the particular algorithms within the program. To a certain extent this chapter may be thought of as introducing the basic conventions and notations to be used later in describing the unit operation algorithms themselves. The actual routines to be documented in this chapter are as follows:

1) CMPBRK, which computes a breakage matrix
2) BRSIZE, which computes a bridge based on size intervals
3) INTS, which performs linear interpolation
4) YINTRP, which performs 4-point Lagrangian interpolation
5) SDSIZE, which summarizes flowstream data by size increment.
6) DSTRB, which splits a flowstream according to a distribution curve
7) ECON, which computes economic data.

4.1 Subroutine CMPBRK

The breakage that takes place during the operation of a unit is simulated by subroutine BREAK2. The breakage is characterized in part by a breakage matrix which is computed by this utility CMPBRK. Following Broadbent and Callcott(3,4) the breakage matrix is obtained from a breakage function and the grade sizes of the feed. The breakage function is defined as

$$B(x,y) = \frac{(1 - \exp(-x/y))}{(1 - \exp(-1))}$$

where B(x,y) is the fraction of coal in initial size y broken below size x. The i-jth element of the breakage matrix is then the proportion of feed which was in the jth size grade before breakage and is in the ith size grade after breakage. To allow for more severe breakage, integral powers ($\geq$ 1) of the breakage matrix may be used. The computation of the breakage matrix raised to the desired power is performed by subroutine CMPBRK. Figure II-4-1.1 gives the listing and basic documentation for this routine.

```
      SUBROUTINE CMPBRK(BR,TEMP,KBR,SIZE,NSIZE)
C
C     TITLE(COMPUTE BREAKAGE MATRIX)
C
C
C     SUBROUTINE CMPBRK COMPUTES THE BREAKAGE MATRIX RAISED TO THE
C     KBR-TH POWER.
C
C     PARAMETER       DESCRIPTION
C     BR              THIS 2-DIMENSIONAL UPPER TRIANGULAR REAL ARRAY IS
C                     USED TO CONTAIN THE PROPORTION OF COAL IN ONE SIZE
C                     FRACTION THAT WILL BREAK INTO ANOTHER SIZE FRACTION.
C     TEMP            THIS 2-DIMENSIONAL REAL ARRAY IS USED FOR SCRATCH
C                     STORAGE.
C     KBR             THIS INTEGER VARIABLE CONTAINS THE POWER OF THE
C                     BREAKAGE MATRIX TO BE CALCULATED.
C     SIZE            THIS REAL 1-DIMENSIONAL ARRAY CONTAINS THE SIZE
C                     DISTRIBUTION DEFINITION FOR THE FLOWS.
C     NSIZE           THIS INTEGER VARIABLE CONTAINS THE NUMBER OF SIZE
C                     FRACTIONS DEFINING THE FLOWS.
C
C     EXTERNAL        EXP
C
C     VARIABLE        DESCRIPTION
C     I,J,K           THESES DUMMY INTEGER VARIABLES ARE USED TO CONTROL
C                     THE MOVEMENT THROUGH MISCELLANEOUS LOCAL LOOPS.
C     SMID            THIS 1-DIMENSIONAL REAL ARRAY CONTAINS THE MID-POINTS
C                     OF THE SIZE FRACTIONS.
C     L               THIS DUMMY INTEGER VARIABLE IS USED AS A LOWER
C                     BOUND FOR LOOPS PROCESSING THE TRIANGULAR BREAKAGE
C                     ARRAY.
C     NS              THIS INTEGER VARIABLE CONTAINS 1 LESS THAN THE
C                     NUMBER OF SIZE FRACTIONS CLASSIFYING THE FLOWSTREAMS.
C     IK              THIS INTEGER VARIABLE CONTAINS THE POWER OF THE
C                     BREAKAGE MATRIX MOST RECENTLY EVALUATED.
C     RATIOB          THIS REAL VARIABLE IS USED TO CONTAIN THE ARGUMENT
C                     FOR THE BREAKAGE FUNCTION.  IT CONTAINS THE RATIO
C                     OF THE SIZES.
C
      DIMENSION BR(NSIZE,NSIZE),TEMP(NSIZE,NSIZE),SIZE(NSIZE)
      COMMON/LOCALV/SMID(23),L,NS,IK,LVFILL(169)
C
C     STEP 1          COMPUTE THE MID-POINTS OF THE SIZE FRACTIONS.
C
      DO 10 I=1,NSIZE
      SMID(I)=(SIZE(I)+SIZE(I+1))/2.0
   10 CONTINUE
C
C     STEP 2          USING THE GENERAL BREAKAGE EQUATION COMPUTE THE
```

FIGURE II-4-1.1. LISTING OF SUBROUTINE CMPBRK

```
C                    CUMULATIVE BREAKAGE MATRIX.
C
      DO 20 I=1,NSIZE
      BR(I,I)=1.0
      IF(I.EQ.NSIZE) GOTO 20
      L=I+1
      DO 15 J=L,NSIZE
      RATIOB=SIZE(J)/SMID(I)
      BR(I,J)=(1.0-EXP(-RATIOB))/0.63214
   15 CONTINUE
   20 CONTINUE
C
C     STEP 3        COMPUTE THE FIRST POWER OF THE NON-CUMULATIVE
C                   BREAKAGE MATRIX.
C
      NS=NSIZE-1
      DO 25 I=1,NS
      DO 25 J=I,NS
      BR(I,J)=BR(I,J)-BR(I,J+1)
   25 CONTINUE
C
C     STEP 4        IF THE REQUIRED POWER OF THE BREAKAGE MATRIX IS 1,
C                   END LOCAL PROCESSING.  ELSE SET THE CURRENT POWER OF
C                   THE BREAKAGE MATRIX TO 1.
C
      IF(KBR.EQ.1) GOTO 55
      IK=1
C
C     STEP 5        PUT THE FIRST POWER OF THE BREAKAGE MATRIX IN THE
C                   UPPER TRIANGLE OF THE TEMPORARY STORAGE ARRAY.
C
      DO 30 I=1,NSIZE
      DO 30 J=I,NSIZE
      TEMP(I,J)=BR(I,J)
   30 CONTINUE
C
C     STEP 6        COMPUTE THE NON-DIAGONAL ELEMENTS OF THE PRODUCT
C                   OF THE FIRST POWER OF THE BREAKAGE MATRIX WITH THE
C                   CURRENT POWER OF THE BREAKAGE MATRIX.  STORE THE
C                   RESULT IN THE LOWER TRIANGLE OF THE TEMPORARY
C                   STORAGE ARRAY.
C
   35 CONTINUE
      DO 40 I=1,NSIZE
      L=I+1
      DO 40 J=L,NSIZE
      TEMP(J,I)=0.0
      DO 40 K=I,J
      TEMP(J,I)=TEMP(J,I)+TEMP(I,K)*BR(K,J)
```

FIGURE II-4-1.1. (Continued)

```
   40 CONTINUE
C
C     STEP 7          STORE THE CURRENT POWER OF THE BREAKAGE MATRIX
C                     IN ARRAY BR.
C
      DO 50 I=1,NSIZE
      L=I+1
      DO 45 J=L,NSIZE
      BR(I,J)=TEMP(J,I)
   45 CONTINUE
      BR(I,I)=TEMP(I,I)*BR(I,I)
   50 CONTINUE
C
C     STEP 8          INCREASE THE CURRENT POWER OF THE BREAKAGE MATRIX
C                     BY 1.
C
      IK=IK+1
C
C     STEP 9          IF THE CURRENT POWER OF THE BREAKAGE MATRIX IS THE
C                     REQUIRED POWER, END LOCAL PROCESSING.  ELSE LOOP
C                     BACK TO STEP 6.
C
      IF(KBR.NE.IK) GOTO 35
   55 CONTINUE
      RETURN
      END
```

FIGURE II-4-1.1. (Continued)

4.2 Subroutine BRSIZE

In Section I it was pointed out that the program assumes that characteristics are uniformly distributed over their size and specific gravity fraction classifications. Subroutine BRSIZE is the core utility routine which implements this assumption as it relates to the size fractions.

The most important, and unfortunately complicated, ramification of the uniformity assumption has to do with its interaction with the size fraction classification schemes. Figure II-4-2.1 shows the values for percent ash classified by size for a typical flow stream. Notice that for any given size fraction interval in terms of which the flow is classified, all sizes within it show the same percent ash. This follows directly from the uniformity assumption. If, for example, an estimate were needed for the fraction 1 inch to .9 inch, which falls entirely within the 1-1/4 to 3/4 inches fraction, the value would simply be 45 percent.

The complication arises when an estimate is needed for some size fraction that does not fall entirely within a defined fraction. As an example, suppose an estimate of percent ash were needed from the data shown in Figure II-4-2.1 for the size fraction 1 inch to .6 inch. One portion of this fraction (1 inch to 3/4 inch) is shown as having 45 percent ash, while the other portion (3/4 inch to .6 inch) has 34.6 percent ash. The appropriate calculation then would be to take the weighted average of the two values. Consider that

$$(1.0 - .75)/(1.0 - .6) = .625 \tag{1}$$

represents that portion of the material in the interval 1.0 to .6 inch which is also in the interval 1.0 to .75 inch, and

$$(.75 - .6)/(1.0 - .6) = .375 \tag{2}$$

represents that portion of the material in the .75 to .6 interval. These follow from assuming that all material is uniformly distributed within the size fractions. Therefore,

$$.675 \times 45 + .375 \times 34.6 = 43.35 \tag{3}$$

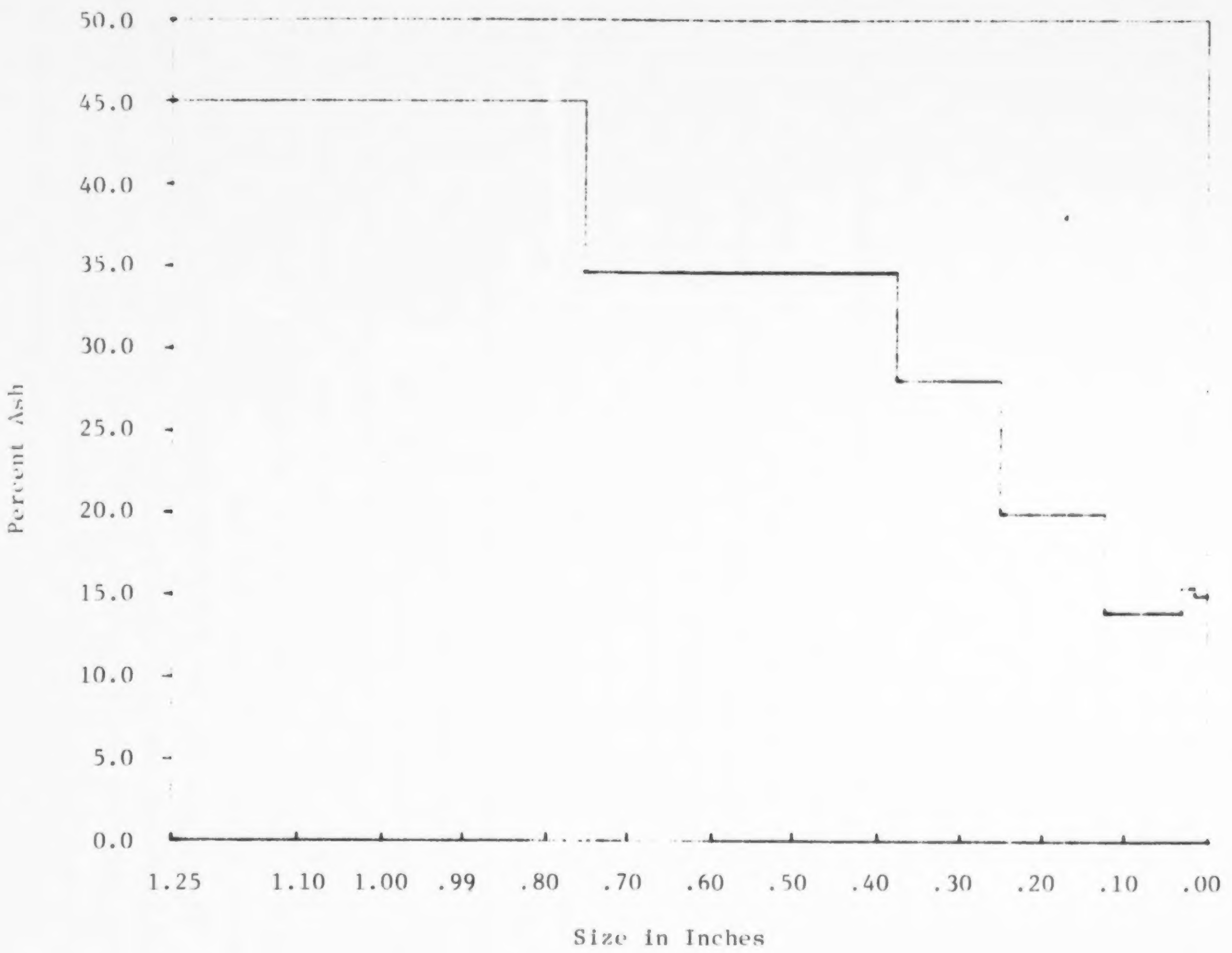


FIGURE II-4-2.1. PERCENT ASH BY SIZE

is the appropriate estimate for the percent ash for the 1 to .6 inch size fraction.

The above example does not represent an actual calculation needed in the program and was chosen only as an illustration. At no time is it necessary to estimate coal characteristics of the flows for size fraction classifications other than the one over which they are defined. In fact, the inverse is needed. All of the following algorithms define size fraction distributions based on their controlling paramaters.

(1) Single-stage washer
(2) Two-stage washer
(3) Forth flotation cell
(4) Rotary breaker
(5) Crusher
(6) Screen

These unit defined size distributions are completely independent of the size distribution used to define the flows in the plant.

The unit distributions are used not just to determine what characteristics certain size fractions have, but also to determine what particular mathematical algorithm is used to apply to that size fraction. As an example, consider the screening algorithm that might be simulating the passage of coal in the size ranges 1/4 to 1/8 inch, 1/8 inch to 10 mesh, and 10 to 28 mesh over an 8-mesh screen. The screening algorithm assumes that all material larger than the screen size will pass over, and that some proportion of the material smaller than the screen size will pass through. Within the screening simulation, then, there are two algorithms which might be applied - the pass over algorithm and the pass through algorithm. The pass over algorithm would apply to the entire fraction 1/4 and 1/8 inch and to that portion of the 1/8 to 10 mesh fraction which is above 8 mesh. The pass through algorithm would apply to that portion of the 1/8 to 10 mesh fraction below 8 mesh and to the entire 10 mesh to 28 mesh fraction.

Within the routines simulating the above listed unit operations, there are minimally two sources of complexity. One relates to implementing the various fraction specific algorithms and the other relates to determining which portions of which size fractions should have which algorithms applied to them. Within program CPSM4, these two operations have been split.

A single subroutine BRSIZE was written to perform the estimation needed to reconstitute one set of size fractions into another. This subroutine is described in detail in this part. The separation of the redistribution logic from the remaining simulation logic in the unit operation routines makes those routines much easier to work with.

Reiterating an earlier discussion, a ubiquitous complication within program CPSM4, as in most programs which simulate real world situations, is that data, coefficients, and/or algorithms predicated on some classification cross some characteristic. Algorithms which perform or support such reclassification operations are termed bridging algorithms. In CPSM4 information and algorithms classified by arbitrary size fraction must be applied to the size fraction classification scheme used to define the flows; therefore, a bridging algorithm is needed to bridge from one size classification to another. Note that the characteristic to be classified is always the same -- the size of coal. Subroutine BRSIZE contains this size bridging algorithm. Figure II-4-2.2 is the flowchart for subroutine BRSIZE while Figure II-4-2.3 presents the FORTRAN listing.

The basic assumptions made within BRSIZE are as follows. First, all size classifications are defined as a monotone decreasing sequence of n+1 non-negative reals, where n is the number of fractions defined by the classification. Let

$$s = s_1, \ldots, s_n, s_{n+1} \tag{4}$$

be such a sequence. By definition then, the i-th size fraction contains no coal larger than s_i and no coal smaller than or equal to s_{i+1}. Saying this differently, the sequence s can be said to induce a definition of n closed-open, ordered intervals

$$[s_1, s_2), [s_2, s_3), \ldots, [s_n, s_{n+1}) \tag{5}$$

which define n mutually exclusive, dense size fractions. These fractions are defined from largest to smallest.

Second, the fact that bridging is needed implies that there are two classifications:

SI = $si_1,\ldots,si_{n+1}$, the initial classification over which the information or algorithm is defined; and

SF = $sf_1,\ldots,sf_{nf+1}$, the final classification to which the formation or algorithm is to be applied.

Since all size fractions in SF must be treated, the following relationship must hold between SI and SF.

$$si_1 \geq sf_1 > sf_{nf+1} \geq si_{ni+1} \tag{6}$$

This relationship simply requires that all sizes encompassed by the final classification must be encompassed by the initial classification; though the initial classification may encompass more. For most initial classifications generated within the program the following holds:

$$\begin{aligned} &si_1=100 \text{ inches} \\ &si_{ni+1}=0 \text{ inches.} \end{aligned} \tag{7}$$

Thus, the program assumes that no flows containing coal particles larger than 100 inches will be simulated. The figure 100 inches was, of course, chosen to be arbitrarily large.

As was implied in the previous part, the definition of the two size classifications is all that is needed to determine the bridge. This fact follows directly from the uniformity hypothesis. To describe how the calculation proceeds requires introducing some additional terminology. Let $[si_j,si_{j+1})$ be an arbitrary fraction within the initial size classification and let $[sf_k,sf_{k+1})$ be an arbitrary fraction in the final size classification. Now if

$$\begin{aligned} &si_j \geq sf_k \geq si_{j+1} \quad \text{and/or} \\ &si_j \geq sf_{k+1} \geq si_{j+1} \quad \text{and/or} \\ &sf_k \geq si_j > si_{j+1} \geq sf_{k+1} \end{aligned} \tag{8}$$

then some part of $[sf_k,sf_{k+1})$ is contained within $[si_j,si_{j+1})$. In this case, we will say that $[si_j,si_{j+1})$ contributes to $[sf_k,sf_{k+1})$. What is to be calculated is that proportion of $[sf_k,sf_{k+1})$ to which $[si_j,si_{j+1})$ contributes. This will be termed the contribution ratio cr.

As shown in an earlier example, the denominator of the contribution ratio is always the width of the final size fraction. For the general case here, it is $sf_k - sf_{k+1}$. The numerator for cr is calculated in one of four ways.

$$cr = \begin{cases} (sf_k - sf_{k+1})/(sf_k - sf_{k+1}) = 1.0 \text{ if } si_j, \geq sf_k > sf_{k+1} \geq si_{j+1} \\ (sf_k - si_{j+1})/(sf_k - sf_{k+1}) < 1.0 \text{ if } si_j \geq sf_k \geq si_{j+1} > sf_{k+1} \\ (si_j - sf_{k+1})/(sf_k - sf_{k+1}) < 1.0 \text{ if } sf_k \geq sf_j \geq sf_{k+1} > si_{j+1} \\ (si_j - si_{j+1})/(sf_k - sf_{k+1}) < 1.0 \text{ if } sk_k \geq si_j > si_{j+1} \geq sf_{k+1} \end{cases} \quad (9)$$

Given the above, the sum of all the contributions to a given final size fraction will always be one. The proof, though straightforward, is lengthy in its details and will not be given here.

The basic information returned by subroutine BRSIZE is then the set of contribution ratios. These are returned in a typeless array to be described in the basic documentation. This array is termed the _bridge specification array_.

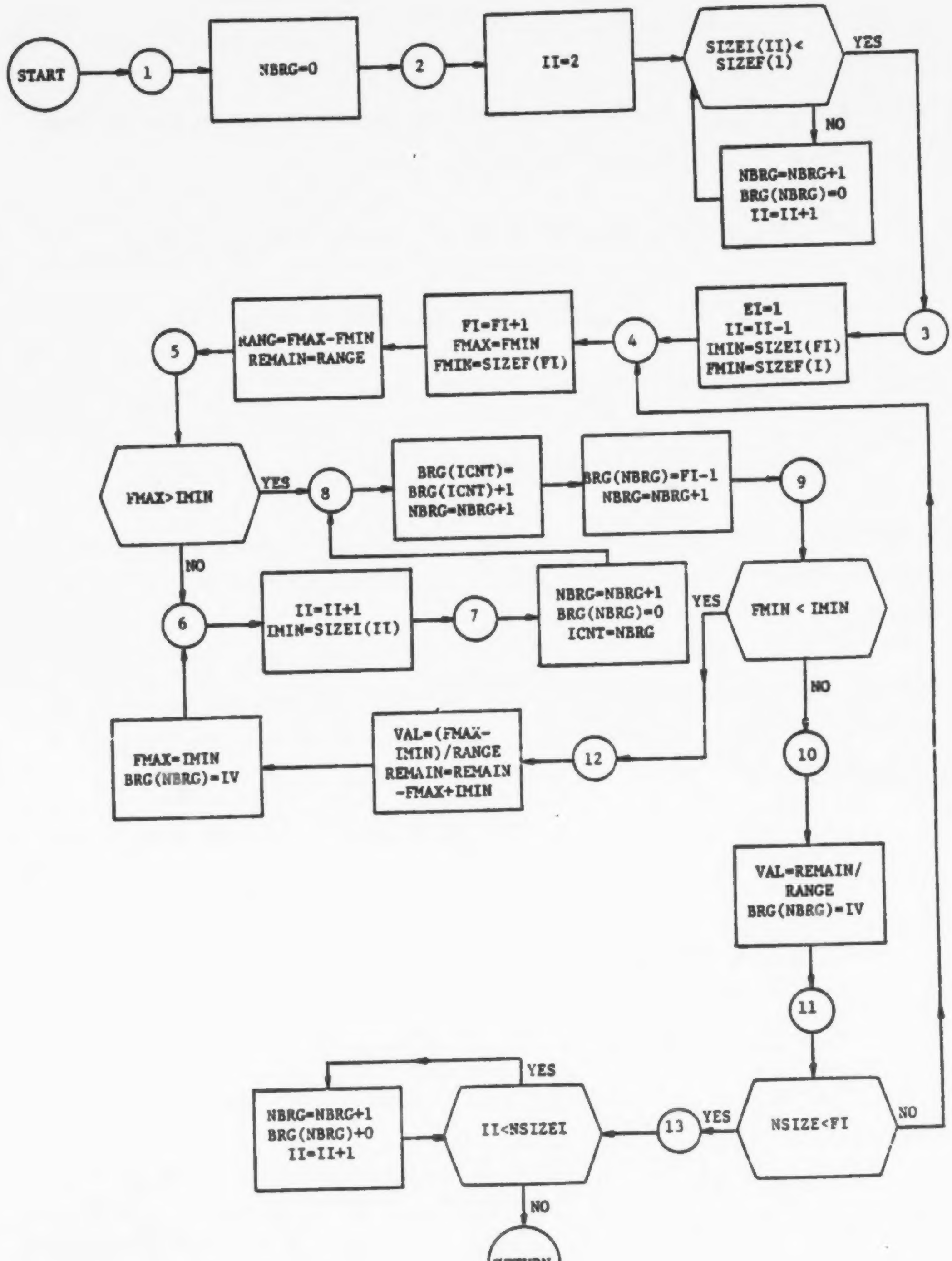

START
1
NBRG=0
2
II=2
SIZEI(II)< SIZEF(1)
YES
NO
NBRG=NBRG+1
BRG(NBRG)=0
II=II+1
3
EI=1
II=II-1
IMIN=SIZEI(FI)
FMIN=SIZEF(I)
4
FI=FI+1
FMAX=FMIN
FMIN=SIZEF(FI)
RANG=FMAX-FMIN
REMAIN=RANGE
5
FMAX>IMIN
YES
8
BRG(ICNT)=
BRG(ICNT)+1
NBRG=NBRG+1
BRG(NBRG)=FI-1
NBRG=NBRG+1
9
NO
6
II=II+1
IMIN=SIZEI(II)
7
NBRG=NBRG+1
BRG(NBRG)=0
ICNT=NBRG
YES
FMIN < IMIN
NO
FMAX=IMIN
BRG(NBRG)=IV
VAL=(FMAX-
IMIN)/RANGE
REMAIN=REMAIN
-FMAX+IMIN
12
10
VAL=REMAIN/
RANGE
BRG(NBRG)=IV
11
YES
NBRG=NBRG+1
BRG(NBRG)+0
II=II+1
II<NSIZEI
13
YES
NSIZE<FI
NO
NO
RETURN

FIGURE II-4-2.2. FLOWCHART FOR SUBROUTINE BRSIZE

Reproduced from
best available copy.

```
      SUBROUTINE BRSIZE(SIZEI,NSIZEI,SIZEF,NSIZEF,BRG,NBRG)
C
C     TITLE(COMPUTE BRIDGE BASED ON SIZE INTERVALS)
C
C     SUBROUTINE BRSIZE IS USED TO COMPUTE THE BRIDGE NEEDED TO ALLO-
C     CATE FLOW DATA SPECIFIED IN ONE SET OF SIZE INTERVALS (THE INI-
C     TIAL INTERVALS) INTO ANOTHER SET OF SIZE INTERVALS (THE FINAL
C     INTERVALS). THIS ROUTINE MAKES THE ASSUMPTIONS THAT BOTH INTERVALS
C     ARE DEFINED IN DESCENDING ORDER AS CLOSED-OPEN INTERVALS AND THAT
C     THE SIZE RANGE DEFINED BY THE FINAL INTERVALS COMPLETELY ENCOM-
C     PASSES THE RANGE DEFINED BY THE INITIAL INTERVALS.
C
C     PARAMETER     DESCRIPTION
C     SIZEI(1)      THIS REAL 1-DIMENSIONAL MONOTONE-DECREASING ARRAY
C                   DEFINES THE INITIAL SIZE INTERVALS.
C     NSIZEI        THIS INTEGER VARIABLE DEFINES THE NUMBER OF INITIAL
C                   SIZE INTERVALS.
C     SIZEF(1)      THIS REAL 1-DIMENSIONAL MONOTONE-DECREASING ARRAY
C                   DEFINES THE FINAL SIZE INTERVALS.
C     NSIZEF        THIS INTEGER VARIABLE DEFINES THE NUMBER OF FINAL
C                   SIZE INTERVALS.
C     BRG(1)        THIS TYPELESS 1-DIMENSIONAL ARRAY CONTAINS THE
C                   BRIDGE SPECIFICATION. IT CONTAINS ONE VARIABLE LENGTH
C                   RECORD FOR EACH FINAL SIZE INTERVAL. EACH OF THESE
C                   RECORDS IS STRUCTURED AS FOLLOWS.
C                   CELL       DESCRIPTION
C                   0          THE NUMBER OF INITIAL SIZE INTERVALS THAT
C                              CONTRIBUTE TO THE FINAL SIZE INTERVAL.
C                              THIS CELL MAY CONTAIN ZERO.
C                   2*I-1      THE SEQUENCE NUMBER OF THE I-TH INITIAL
C                              INTERVAL CONTRIBUTING TO THE FINAL INTERVAL
C                   2*I        THE CONTRIBUTION RATIO FOR THE I-TH
C                              INTERVAL TO THE FINAL INTERVAL.
C     NBRG          THIS INTEGER VARIABLE RETURNS THE NUMBER OF CELLS
C                   USED IN THE BRIDGE SPECIFICATION ARRAY.
C
C     VARIABLE      DESCRIPTION
C     FI            THIS INTERGER VARIABLE IS USED TO CONTROL THE MOVE-
C                   MENT THROUGH THE FINAL INTERVALS. IT NORMALLY POINTS
C                   TO THE MINIMUM CELL OF THE CURRENT FINAL INTERVAL.
C                   THUS, IT IS EQUAL TO ONE PLUS THE INTERVAL NUMBER.
C     FMIN          THIS REAL VARIABLE CONTAINS THE MINIMUM OF THE CUR-
C                   RENT FINAL INTERVAL.
C     ICNT          THIS INTEGER VARIABLE CONTAINS THE ADDRESS IN THE
C                   BRIDGE SPECIFICATION ARRAY OF THE COUNT CELL FOR
C                   THE CURRENT FINAL INTERVAL.
C     II            THIS INTEGER VARIABLE CONTROLS THE MOVEMENT THROUGH
C                   THE INITIAL INTERVALS. IT NORMALLY POINTS TO THE
C                   CELL CONTAINING THE MINIMUM FOR THE INTERVAL. THUS,
```

FIGURE II-4-2.3. LISTING FOR SUBROUTINE BRSIZE

```
C                    IT EQUALS ONE PLUS THE INTERVAL NUMBER.
C     IMAX           THIS REAL VARIABLE CONTAINS THE MAXIMUM OF THE
C                    CURRENT INITIAL INTERVAL RELATIVE TO THE CURRENT
C                    FINAL INTERVAL.
C     IMIN           THIS REAL VARIABLE CONTAINS THE MINIMUM OF THE
C                    CURRENT INITIAL INTERVAL.
C     IV,VAL         THIS TYPELESS VARIABLE CONTAINS THE VALUE OF THE
C                    CONTRIBUTION FRACTION OF THE CURRENT INITIAL INTER-
C                    VAL TO THE CURRENT FINAL INTERVAL STORED IN FLOATING
C                    POINT FORM.
C     RANGE          THIS REAL VARIABLE CONTAINS THE VALUE OF THE OVER-
C                    ALL RANGE OF THE CURRENT INITIAL INTERVAL.
C     REMAIN         THIS REAL VARIABLE CONTAINS THE AMOUNT OF THE CUR-
C                    RENT INITIAL INTERVAL REMAINING WHICH HAS NOT YET
C                    CONTRIBUTED TO A FINAL INTERVAL.
C
      DIMENSION SIZEI(1),SIZEF(1),BRG(1)
      INTEGER BRG,FI
      REAL IMIN,IMAX
      EQUIVALENCE (VAL,IV)
C
C     STEP 1         INITIALIZE THE NUMBER OF CELLS USED IN THE BRIDGE
C                    ARRAY AT ZERO.
C
      NBRG=0
C
C     STEP 2         MOVE FORWARD IN THE FINAL INTERVALS UNTIL THAT INTER-
C                    VAL CONTAINING THE START OF THE FIRST INITIAL INTERVAL
C                    IS FOUND. SET THE NUMBER OF INITIAL INTERVALS CONTRI-
C                    BUTING TO THE FINAL INTERVALS TO ZERO FOR THOSE
C                    FINAL INTERVALS WHOSE ENTIRE RANGE EXCEEDS THE START
C                    OF THE FIRST INITIAL INTERVAL.
C
      FI=2
   10 IF(SIZEF(FI).LT.SIZEI(1))GO TO 15
      NBRG=NBRG+1
      BRG(NBRG)=0
      FI=FI+1
      GO TO 10
C
C     STEP 3         THAT FINAL INTERVAL WHICH CONTAINS THE FIRST INI-
C                    IAL INTERVAL HAS BEEN FOUND. INITIALIZE THE BASIC
C                    CONTROL VARIABLES WHICH WILL CONTROL THE MOVEMENT
C                    THROUGH THE TWO SETS OF INTERVALS.
C
   15 II=1
      FI=FI-1
      FMIN=SIZEF(FI)
      IMIN=SIZEI(1)
```

FIGURE II-4-2.3. (Continued)

```
C
C     STEP 4        THE CURRENT INITIAL INTERVAL ENDED WITHIN THE
C                   CURRENT FINAL INTERVAL. RESET THE INITIAL INTERVAL
C                   SO THAT THE CURRENT ONE NOW BEGINS WITHIN THE FINAL
C                   INTERVAL.
C
   20 II=II+1
      IMAX=IMIN
      IMIN=SIZEI(II)
      RANGE=IMAX-IMIN
      REMAIN=RANGE
C
C     STEP 5        IF THE START OF THE INITIAL INTERVAL EXCEEDS THE END
C                   OF THE CURRENT FINAL INTERVAL THEN IT IS WITHIN THE
C                   FINAL INTERVAL. IN THIS CASE, BRANCH TO STEP 8.
C                   NOTE THAT IT IS AT THIS POINT THAT THE CLOSED-OPEN
C                   INTERVAL ASSUMPTION IS USED.
C
      IF(IMAX-FMIN)25,25,30
C
C     STEP 6        THE INITIAL INTERVAL IS NOT WITHIN THE CURRENT FINAL
C                   INTERVAL. INCREMENT THE CURRENT FINAL INCREMENT
C                   CONTROL VARIABLES.
C
   25 FI=FI+1
      FMIN=SIZEF(FI)
C
C     STEP 7        INCREMENT THE NUMBER OF CELLS USED IN THE BRIDGE
C                   SPECIFICATION ARRAY BY ONE TO ALLOW FOR THE COUNT
C                   CELL FOR THE CURRENT FINAL INTERVAL AND INITIALIZE
C                   THAT CELL AT ZERO.
C
      NBRG=NBRG+1
      BRG(NBRG)=0
      ICNT=NBRG
C
C     STEP 8        AT THIS POINT IT IS KNOWN THAT THE CURRENT INITIAL
C                   INTERVAL CONTRIBUTES TO THE CURRENT FINAL INTERVAL.
C                   INCREMENT THE NUMBER OF INITIAL INTERVALS CONTRIBUT-
C                   ING TO THIS FINAL INTERVAL BY ONE AND STORE THE
C                   SEQUENCE NUMBER OF THE INITIAL INTERVAL IN THE
C                   NEXT AVAILABLE CELL OF THE BRIDGE SPECIFICATION
C                   ARRAY.
C
   30 BRG(ICNT)=BRG(ICNT)+1
      NBRG=NBRG+1
      BRG(NBRG)=II-1
      NBRG=NBRG+1
C
```

FIGURE II-4-2.3. (Continued)

```
C     STEP 9          IF THE MINIMUM OF THE CURRENT INITIAL INTERVAL IS
C                     LESS THAN THE MINUMUM OF THE CURRENT FINAL INTERVAL,
C                     BRANCH TO STEP 12.
C
      IF(IMIN.LT.FMIN)GO TO 35
C
C     STEP 10         THE INITIAL INTERVAL ENDS WITHIN THE FINAL INTERVAL,
C                     THEREFORE, THE ENTIRE REMAINDER CONTRIBUTES TO THAT
C                     FINAL INTERVAL. COMPUTE THE FRACTION AND STORE IT
C                     IN THE NEXT AVAILABLE CELL OF THE BRIDGE SPECIFICA-
C                     TION ARRAY.
C
      VAL=REMAIN/RANGE
      BRG(NBRG)=IV
C
C     STEP 11         IF ALL INITIAL INTERVALS HAVE BEEN PROCESSED,
C                     BRANCH TO STEP 13. ELSE, LOOP BACK TO STEP 4.
C
      IF(NSIZEI-II)40,20,20
C
C     STEP 12         THE CURRENT INITIAL INTERVAL EXTENDS BEYOND THE
C                     RANGE OF THE CURRENT FINAL INTERVAL. COMPUTE THE
C                     FRACTION CONTRIBUTING TO THE FINAL INTERVAL, STORE
C                     IT IN THE NEXT AVAILABLE CELL OF THE BRIDGE SPECIFI-
C                     CATION ARRAY, RECOMPUTE THE INITIAL INTERVAL BOUNDARY
C                     VALUES, AND LOOP BACK TO STEP 6.
C
   35 VAL=(IMAX-FMIN)/RANGE
      REMAIN=REMAIN-IMAX+FMIN
      IMAX=FMIN
      BRG(NBRG)=IV
      GO TO 25
C
C     STEP 13         ALL INITIAL INTERVALS HAVE BEEN PROCESSED. SET THE
C                     NUMBER CONTRIBUTING TO ANY REMAINING FINAL INTER-
C                     VALS TO ZERO, AND END LOCAL PROCESSING.
C
   40 IF(FI.GT.NSIZEF)GO TO 999
      NBRG=NBRG+1
      BRG(NBRG)=0
      FI=FI+1
      GO TO 40
  999 RETURN
      END
```

FIGURE II-4-2.3. (Continued)

4.3 Interpolation and Approximation

In addition to the general topic of conventions and notations, one other general area must be developed before the detailed discussion of the unit operations can begin and before discussing the remaining subroutines. This is the topic of interpolation and approximation. CPSM4 is a mathematical model of the processes within a coal preparation plant. As such it does not actually perform any of the operations within a plant; rather it attempts to maintain the same mathematical relationships between measurable quantities as would result from measuring those quantities within the actual plant. Such relationships are represented mathematically as functions. Unfortunately, it is never possible to represent the real world relationship exactly. This shortcoming arises from a variety of sources such as an incomplete understanding of the phenomena under study, the stocastic or random nature of the relationship, and the uncertainties associated with the very measurements used as basis for establishing the relationship.

Real world relationships are then approximated via functions, based on measured values. The next question to be asked has to do with how the functions are to be represented in the machine. Theoretically, the best representation would be as some exact functional form. Such a representation would seem to be the best since it would always give a precise result for any quantity whose value depends upon the function.

Actually functions of this sort are never programmed in their exact functional form in program CPSM4. Instead they are represented indirectly via a finite set of pairs of values. The normal view of this set of pairs of values is as though they represented the coordinates of points of that function if it were displayed in graphical form. There are two reasons why functions are typically based on measured values. First, their exact form is often not known. Second, different particular unit operations within the various generic types differ only insofar as their functional forms are concerned. It is desirable, then, to be able to vary these functional forms from the data deck of the program. Since all functions of the type of interest here can be represented as a series of points and since such series are very easy to input into the program, all functions are represented via a series of points.

Given that functions are represented as a series of finite points, obtaining a value for any point requires interpolation. The basic concept behind the use of interpolation is that the value of any point of a function can be estimated based upon the values of known points surrounding it. The number of points used and the manner in which particular estimates are computed varies from estimation technique to estimation technique. There is a wide body of literature on this topic; therefore, the various techniques available, their advantages and disadvantages, will not be discussed here.

Within program CPSM4 there are two routines for doing interpolation, subroutine INTS and function YINTRP. Subroutine INTS performs simple linear interpolation; while function YINTRP performs 4-point Lagrangian interpolation.

4.4 Subroutine INTS

The simplest form of interpolation is linear. This type of interpolation bases the dependent value of a given point upon the know dependent values of the two points surrounding it. The assumption made is that the unknown point lies on the straight line connecting the two points. The arithmetic required to implement this assumption is straightforward and is shown in the basic documentation. It should be pointed out that subroutine INTS does not perform extrapolation. Any independent value which lies outside the range specified is assigned the dependent value of the appropriate end point.

Figure II-4-4.1 is a flowchart for subroutine INTS; Figure II-4-4.2 presents the FORTRAN listing.

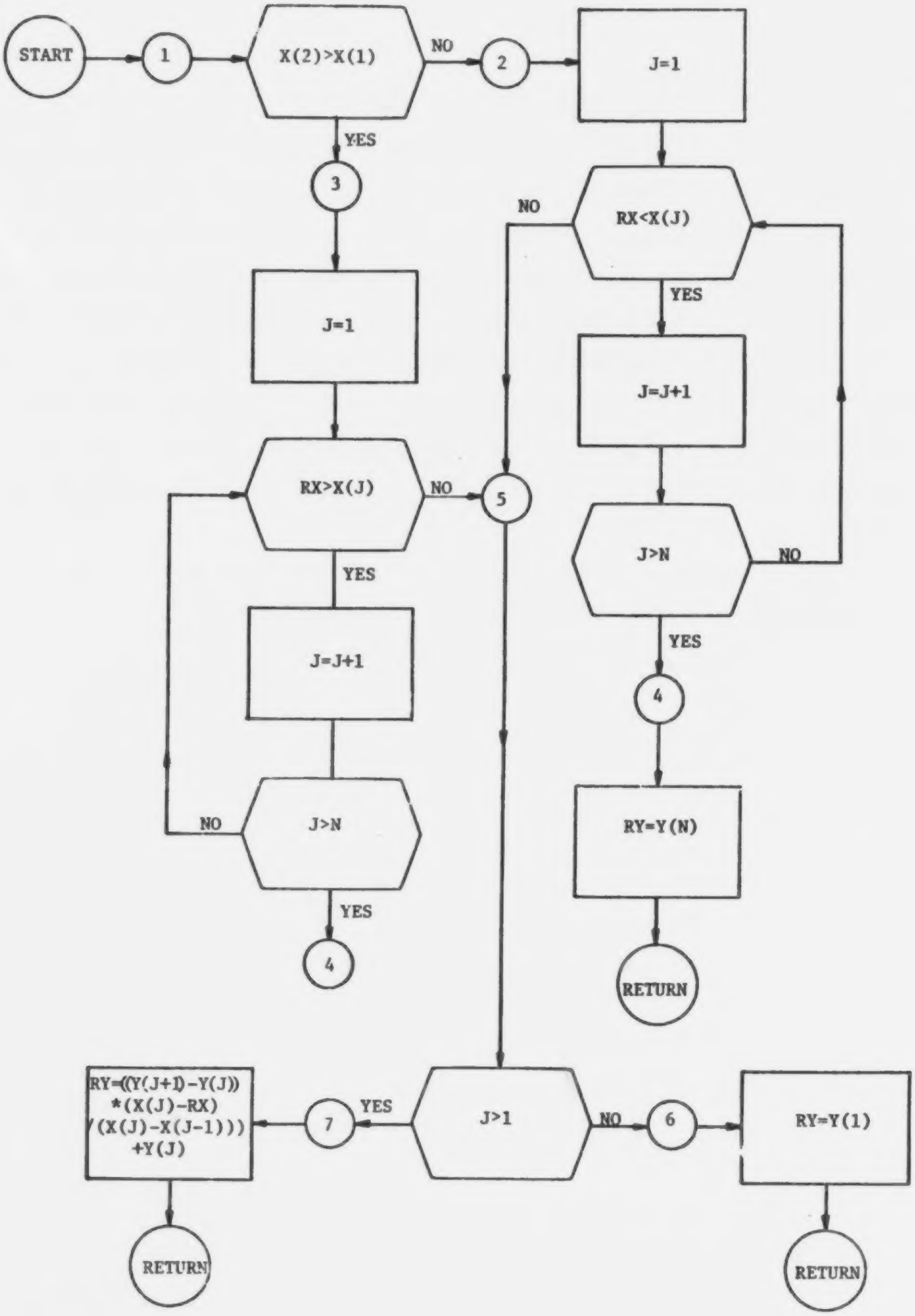

START
1
X(2)>X(1)
NO
2
J=1
YES
3
J=1
NO
RX<X(J)
YES
J=J+1
RX>X(J)
NO
5
YES
J>N
NO
J=J+1
YES
4
NO
J>N
RY=Y(N)
YES
4
RETURN
RY=((Y(J+1)-Y(J))
*(X(J)-RX)
/(X(J)-X(J-1)))
+Y(J)
YES
7
J>1
NO
6
RY=Y(1)
RETURN
RETURN

FIGURE II-4-4.1. FLOWCHART FOR SUBROUTINE INTS

```
      SUBROUTINE INTS(X,Y,RX,RY,N)
C
C     TITLE(PERFORM LINEAR INTERPLOATION)
C
C     SUBROUTINE INTS PERFORMS UNIVARIATE LINEAR INTERPOLATION FROM A
C     TABLE OF VALUES OF X VERSUS Y.
C
C     PARAMETER    DESCRIPTION
C     X            THIS 1-DIMENSIONAL REAL ARRAY CONTAINS THE INDE-
C                  PENDENT X-VARIABLE VALUES. THIS SET OF VALUES MUST
C                  BE EITHER INCREASING OR MONOTONE DECREASING.
C     Y            THIS 1-DIMENSIONAL REAL ARRAY CONTAINS THE DEPENDENT
C                  Y-VARIABLE VALUES.
C     RX           THIS REAL VARIABLE CONTAINS THE VALUE OF THE INDE-
C                  PENDENT VARIABLE AT WHICH A VALUE FOR THE DEPEN-
C                  DENT VARIABLE SHOULD BE COMPUTED.
C     RY           THIS REAL VARIABLE RETURNS THE COMPUTED VALUE FOR THE
C                  DEPENDENT VARIABLE.
C     N            THIS INTEGER VARIABLE DEFINES THE NUMBER OF POINTS
C                  USED TO DEFINE THE VARIABLES.
C
C     VARIABLE     DESCRIPTION
C     J            THIS INTEGER VARIABLE CONTAINS THE SEQUENCE NUMBER
C                  OF THAT POINT IMMEDIATELY TO THE LEFT OF THE VALUE
C                  FOR WHICH THE DEPENDENT VARIABLE IS TO BE COMPUTED.
C                  THIS IS REFERRED TO AS THE BASE POINT.
C
      DIMENSION X(N),Y(N)
C
C     STEP 1       BRANCH ACCORDING TO WHETHER OR NOT THE X-VARIABLE
C                  IS DEFINED AS MONOTONE INCREASING OR DECREASING.
C                  BRANCH    HOW DEFINED
C                  STEP 2    INCREASING
C                  STEP 3    DECREASING
C
      IF(X(N)-X(1))10,10,20
C
C     STEP 2       THE X-VALUES ARE MONOTONE DECREASING. FIND THAT X-
C                  GREATER THAN OR EQUAL TO THE SPECIFIED VALUE. IF ONE
C                  IS FOUND, BRANCH TO STEP 5. ELSE BRANCH TO STEP 4.
C
   10 DO 15 J=1,N
      IF(RX-X(J))15,35,35
   15 CONTINUE
      GO TO 30
C
C     STEP 3       THE X-VALUES ARE MONOTONE INCREASING. FIND THAT X-
C                  VALUE SMALLER THAN OR EQUAL TO THE SPECIFIED VALUE.
C                  IF ONE IS FOUND BRANCH TO STEP 5. ELSE, BRANCH TO
```

FIGURE II-4-4.2. FORTRAN LISTING FOR SUBROUTINE INTS

```
C                       STEP 4.
C
   20 DO 25 J=1,N
      IF(RX-X(J))35,35,25
   25 CONTINUE
C
C     STEP 4            THE SPECIFIED POINT IS BEYOND THE SPECIFIED RANGE OF
C                       THE X-VARIABLE.SET THE DEPENDENT VALUE TO THE LAST
C                       VALUE AND END LOCAL PROCESSING.
C
   30 RY=Y(N)
      GO TO 999
C
C     STEP 5            IF THE SPECIFIED POINT IS WITHIN THE RANGE OF VALUES
C                       DEFINFD, BRANCH TO STEP 6. ELSE SET THE DEPENDENT
C                       VALUE EQUAL TO THE FIRST VALUE, AND END LOCAL PRO-
C                       CESSING.
C
   35 IF(J.GT.1)GO TO 40
      RY=Y(1)
      GO TO 999
C
C     STEP 6            COMPUTE THE VALUE FOR THE DEPENDENT VARIABLE AND
C                       END LOCAL PROCESSING.
C
   40 RY=((Y(J-1)-Y(J))*(X(J)-RX)/(X(J)-X(J-1)))+Y(J)
  999 RETURN
      END
```

FIGURE II-4-4.2. (Continued)

4.5 Function YINTRP

A more complicated form of interpolation than linear is Lagrangian, a form of polynomial interpolation. In particular, function YINTRP bases the dependent value of the given point upon the known dependent values of the four points surrounding it by assuming that the given point lies on a no more than forth degree polynomial connecting the known points. Carnahan, Luther, and Wilkes(1) contains an excellent discussion of Lagrange's interpolation polynomial. Readers unfamiliar with this type of interpolation are referred to that source. The discussion will not be repeated here.

The method of solution used in this implementation is derived and, therefore, will be discussed briefly. The basic formula for Lagrange's 4-point interpolating polynomial is shown in (1).

$$f(x) = \sum_{i=min}^{min+3} \left(\prod_{\substack{j=min \\ j\neq i}}^{min+3} (x-x_j)/(x_i-x_j)\cdot Y_i \qquad (1)$$

By rewriting this as (2) about nine multiplications and nine subtractions can be saved, at the expense of four divisions.

$$c = \prod_{j=min}^{min+3} (x-x_j)$$

$$f(x) = \sum_{i=min}^{min+3} ((C/(x-x_i))/\prod_{\substack{j=min \\ j\neq i}}^{min+3} (x_i-x_j))\cdot Y_i \qquad (2)$$

This is the form used in Function YINTRP.

If any independent values lie outside the range specified, then the dependent value is assigned the value of the appropriate end point; thus, YINTRP performs no extrapolation. If any independent values lie so close to one of the end points that they cannot be surrounded by four points, then linear interpolation is performed instead. Normally the known x-coordinates must be specified in monotone increasing form. The only exception is that all x-points from a certain position onward may have the same value. In these cases, the first point in such a sequence is considered to define the endpoint.

Figure II-4-5.1 is the flowchart for function YINTRP, while Figure II-4-5.2 presents the FORTRAN listing.

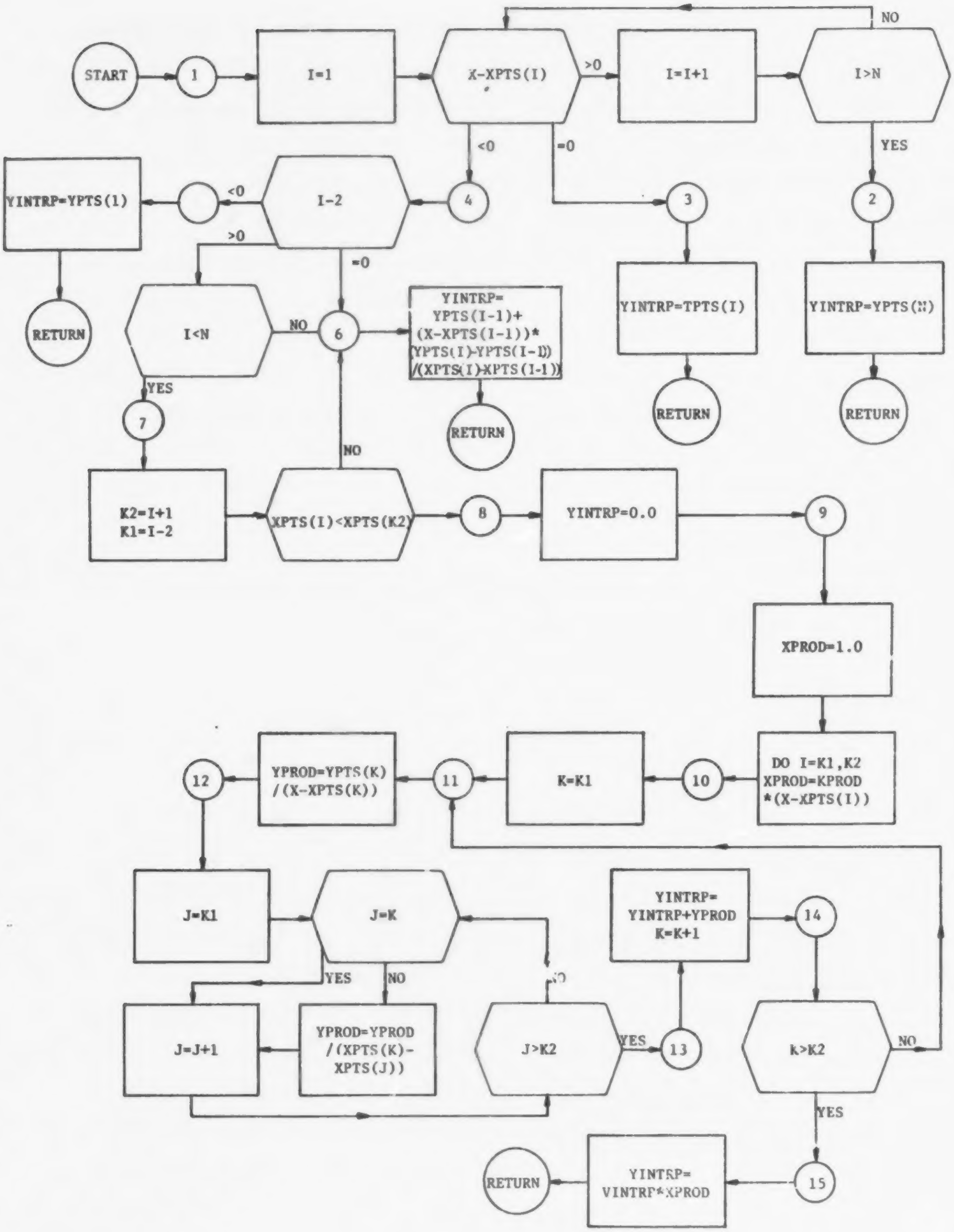
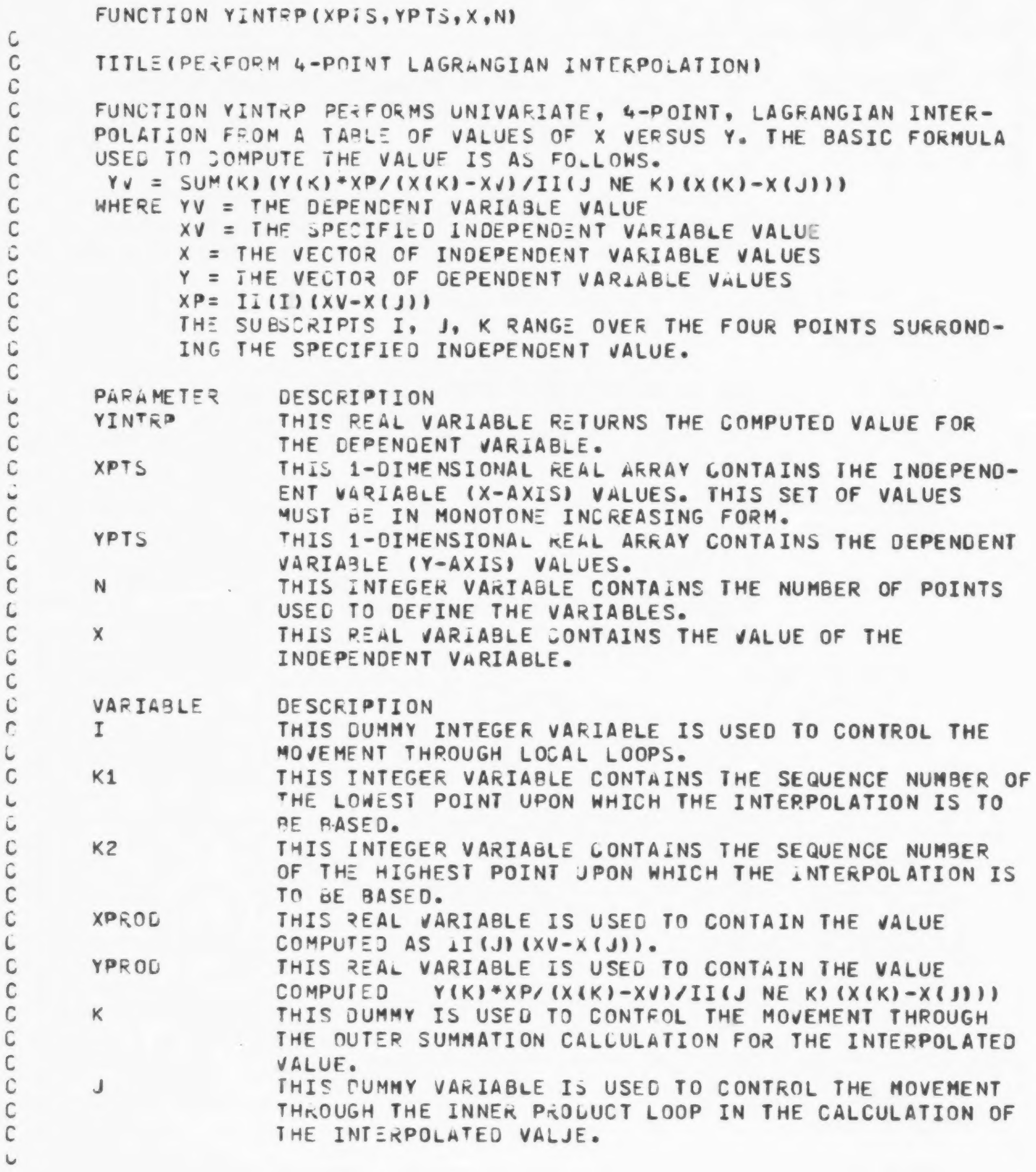

START
1
I=1
X-XPTS(I)
>0
I=I+1
I>N
NO
YES
<0
=0
YINTRP=YPTS(1)
<0
I-2
4
3
2
>0
=0
RETURN
I<N
NO
6
YINTRP=
YPTS(I-1)+
(X-XPTS(I-1))*
(YPTS(I)-YPTS(I-1))
/(XPTS(I)-XPTS(I-1))
YINTRP=TPTS(I)
YINTRP=YPTS(N)
YES
7
RETURN
RETURN
RETURN
NO
K2=I+1
K1=I-2
XPTS(I)<XPTS(K2)
8
YINTRP=0.0
9
XPROD=1.0
12
YPROD=YPTS(K)
/(X-XPTS(K))
11
K=K1
10
DO I=K1,K2
XPROD=KPROD
*(X-XPTS(I))
J=K1
J=K
NO
YINTRP=
YINTRP+YPROD
K=K+1
14
YES
NO
J=J+1
YPROD=YPROD
/(XPTS(K)-
XPTS(J))
J>K2
YES
13
k>K2
NO
YES
RETURN
YINTRP=
VINTRP*XPROD
15

FIGURE II-4-5.1. FLOWCHART FOR FUNCTION YINTRP



```
      FUNCTION YINTRP(XPTS,YPTS,X,N)
C
C     TITLE(PERFORM 4-POINT LAGRANGIAN INTERPOLATION)
C
C     FUNCTION YINTRP PERFORMS UNIVARIATE, 4-POINT, LAGRANGIAN INTER-
C     POLATION FROM A TABLE OF VALUES OF X VERSUS Y. THE BASIC FORMULA
C     USED TO COMPUTE THE VALUE IS AS FOLLOWS.
C      YV = SUM(K)(Y(K)*XP/(X(K)-XV)/II(J NE K)(X(K)-X(J)))
C     WHERE YV = THE DEPENDENT VARIABLE VALUE
C           XV = THE SPECIFIED INDEPENDENT VARIABLE VALUE
C           X = THE VECTOR OF INDEPENDENT VARIABLE VALUES
C           Y = THE VECTOR OF DEPENDENT VARIABLE VALUES
C           XP= II(I)(XV-X(J))
C           THE SUBSCRIPTS I, J, K RANGE OVER THE FOUR POINTS SURROND-
C           ING THE SPECIFIED INDEPENDENT VALUE.
C
C     PARAMETER    DESCRIPTION
C     YINTRP       THIS REAL VARIABLE RETURNS THE COMPUTED VALUE FOR
C                  THE DEPENDENT VARIABLE.
C     XPTS         THIS 1-DIMENSIONAL REAL ARRAY CONTAINS THE INDEPEND-
C                  ENT VARIABLE (X-AXIS) VALUES. THIS SET OF VALUES
C                  MUST BE IN MONOTONE INCREASING FORM.
C     YPTS         THIS 1-DIMENSIONAL REAL ARRAY CONTAINS THE DEPENDENT
C                  VARIABLE (Y-AXIS) VALUES.
C     N            THIS INTEGER VARIABLE CONTAINS THE NUMBER OF POINTS
C                  USED TO DEFINE THE VARIABLES.
C     X            THIS REAL VARIABLE CONTAINS THE VALUE OF THE
C                  INDEPENDENT VARIABLE.
C
C     VARIABLE     DESCRIPTION
C     I            THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                  MOVEMENT THROUGH LOCAL LOOPS.
C     K1           THIS INTEGER VARIABLE CONTAINS THE SEQUENCE NUMBER OF
C                  THE LOWEST POINT UPON WHICH THE INTERPOLATION IS TO
C                  BE BASED.
C     K2           THIS INTEGER VARIABLE CONTAINS THE SEQUENCE NUMBER
C                  OF THE HIGHEST POINT UPON WHICH THE INTERPOLATION IS
C                  TO BE BASED.
C     XPROD        THIS REAL VARIABLE IS USED TO CONTAIN THE VALUE
C                  COMPUTED AS II(J)(XV-X(J)).
C     YPROD        THIS REAL VARIABLE IS USED TO CONTAIN THE VALUE
C                  COMPUTED   Y(K)*XP/(X(K)-XV)/II(J NE K)(X(K)-X(J)))
C     K            THIS DUMMY IS USED TO CONTROL THE MOVEMENT THROUGH
C                  THE OUTER SUMMATION CALCULATION FOR THE INTERPOLATED
C                  VALUE.
C     J            THIS DUMMY VARIABLE IS USED TO CONTROL THE MOVEMENT
C                  THROUGH THE INNER PRODUCT LOOP IN THE CALCULATION OF
C                  THE INTERPOLATED VALUE.
C
```

FIGURE II-4-5.2. FORTRAN LISTING FOR YINTRP

```
      DIMENSION XPTS(N),YPTS(N)
C
C     STEP 1        DETERMINE IF THERE IS A X-VARIABLE POINT IMME-
C                   DIATELY GREATER THAN OR EQUAL TO THE SPECIFIED POINT
C                   AND BRANCH ACCORDINGLY.
C
C                   BRANCH    STATUS
C                   STEP 2    ALL POINTS LESS THAN SPECIFIED POINT
C                   STEP 3    DEFINED POINT EQUALS SPECIFIED POINT
C                   STEP 4    A POINT GREATER THAN THE SPECIFIED ONE
C
      DO 10 I=1,N
      IF(X-XPTS(I))20,15,10
   10 CONTINUE
C
C     STEP 2        THE X-VALUE SPECIFIED IS GREATER THAN ALL THE X-
C                   POINTS DEFINED FOR THE VARIABLE. SET THE Y-VALUE
C                   EQUAL TO THE Y-VALUE OF THE LAST POINT AND END LOCAL
C                   PROCESSING.
C
      YINTRP=YPTS(N)
      GO TO 999
C
C     STEP 3        THE SPECIFIED VALUE EQUALS ONE OF THE DEFINED X-
C                   POINTS. SET THE Y-VALUE EQUAL TO THE CORRESPONDING
C                   Y-POINT VALUE.
C
   15 YINTRP=YPTS(I)
      GO TO 999
C
C     STEP 4        THAT X-POINT WHICH IS IMMEDIATELY GREATER THAN THE
C                   SPECIFIED POINT HAS BEEN FOUND. BRANCH ACCORDING TO
C                   WHICH POINT THAT IS.
C                   BRANCH    POINT
C                   STEP 5    FIRST POINT
C                   STEP 6    SECOND POINT OR LAST POINT
C                   STEP 7    ANY OTHER POINT
C
   20 IF(I-2)30,35,25
   25 IF(I-N)40,35,35
C
C     STEP 5        THE FIRST POINT IS GREATER THAN THE SPECIFIED VALUE.
C                   SET THE Y-VALUE EQUAL TO THE FIRST POINT S Y-VALUE.
C                   THEN END LOCAL PROCESSING.
C
   30 YINTRP=YPTS(1)
      GO TO 999
C
C     STEP 6        THE SPECIFIED POINT IS SO CLOSE TO THE ENDS OF THE
```

FIGURE II-4-5.2. (Continued)

```
C                 DEFINED DISTRIBUTION THAT 4 POINTS SUUROUNDING IT
C                 ARE NOT AVAILABLE. USE LINEAR INTERPOLATION TO COM-
C                 PUTE A Y-VALUE AND END LOCAL PROCESSING.
C
   35 YINTRP=YPTS(I-1)+(X-XPTS(I-1))*(YPTS(I)-YPTS(I-1))/(XPTS(I)
     1       -XPTS(I-1))
      GO TO 999
C
C     STEP 7      4-POINT LAGRANGIAN INTERPOLATION CAN BE PERFORMED.
C                 COMPUTE THE SEQUENCE NUMBERS OF THE HIGHEST AND
C                 LOWEST POINTS TO BE USED IN THE CACULATION.
C
   40 K2=I+1
      K1=I-2
      IF(XPTS(I).GE.XPTS(K2))GO TO 35
C
C     STEP 8      INITIALIZE THE Y-VALUE AT ZERO.
C
      YINTRP=0.0
C
C     STEP 9      COMPUTE THE OVERALL PRODUCT OF THE X-VALUE DIFFER-
C                 ENCES.
C
      XPROD=1.0
      DO 45 I=K1,K2
      XPROD=XPROD*(X-XPTS(I))
   45 CONTINUE
C
C     STEP 10     INITIALIZE THE DUMMY COUNTER TO CONTROL THE OUTER
C                 SUMMATION NEEEDED TO COMPUTE THE VALUE.
C
      DO 60 K=K1,K2
C
C     STEP 11     INITIALIZE THE VALUE OF THE INNER PRODUCT AT THE Y-
C                 VALUE DIVIDED BY THE CURRENT X DIFFERENCE.
C
      YPROD=YPTS(K)/(X-XPTS(K))
C
C     STEP 12     COMPUTE THE INNER Y-PRODUCT VALUE.
C
      DO 55 J=K1,K2
      IF(J-K)50,55,50
   50 YPROD=YPROD/(XPTS(K)-XPTS(J))
   55 CONTINUE
C
C     STEP 13     ADD THE CURRENT INNER Y PRODUCT VALUE INTO THE
C                 FINAL COMPUTED VALUE.
C
      YINTRP=YINTRP+YPROD
```

FIGURE II-4-5.2. (Continued)

```
C
C     STEP 14       INCREMENT THE DUMMY COUNTER CONTROLLING THE OUTER
C                   SUMMATION BY ONE, AND IF ALL POINTS HAVE BEEN PRO-
C                   CESSED, LOOP BACK TO STEP 11.
C
   60 CONTINUE
C
C     STEP 15       MULTIPLY THE THE Y-VALUE COMPUTED ABOVE BY THE
C                   PRODUCT OF THE X DIFFERENCES TO OBTAIN THE FINAL
C                   INTERPOLATED VALUE AND END LOCAL PROCESSING.
C
      YINTRP=YINTRP*XPROD
  999 RETURN
      END
```

FIGURE II-4-5.2. (Continued)

4.6 Subroutine SDSIZE

Many of the routines in the program need access to the flow weight and flow characteristics summarized by size fraction and for the composite. This summarization is performed by subroutine SDSIZE. The calculation is straightforward and is discussed adequately in the basic documentation. It should be observed, however, that the values for the ash, total sulfur, and pyritic sulfur are converted from ratio form to percent form.

Figure II-4-6.1 contains the listing for subroutine SDSIZE. No flowchart, other than the verbal one in the listing, is provided since SDSIZE performs minimal branching.

```
      SUBROUTINE SDSIZE(FLOW,SUM,NSIZE,NGRAV,NCHAR,TOTAL)
C
C     TITLE(SUMMARIZE FLOWSTREAM DATA BY SIZE INCREMENT)
C
C     SUBROUTINE SDSIZE IS USED TO CALCULATE SUMMARY DATA ABOUT A
C     FLOWSTREAM BY SIZE INCREMENTS, THUS REMOVING THE SPECIFIC
C     GRAVITY DETAIL FROM THE SUMMARIZATION.
C
C     PARAMETER     DESCRIPTION
C     FLOW          THIS REAL 3-DIMENSIONAL ARRAY CONTAINS THE DETAILED
C                   INFORMATION DESCRIBING THE FLOW STREAM CLASSIFIED
C                   BY SIZE INCREMENT AND SPECIFIC GRAVITY INCREMENT.
C     SUM           THIS REAL 2-DIMENSIONAL ARRAY RETURNS THE SUMMAR-
C                   IZED DATA BY SIZE INCREMENT AND TOTAL DESCRIBING
C                  THE FLOW STREAM.
C     NSIZE         THIS INTEGER VARIABLE CONTAINS THE NUMBER OF SIZE
C                   INCREMENTS.
C     NGRAV         THIS INTEGER VARIABLE CONTAINS THE NUMBER OF
C                   SPECIFIC GRAVITY INCREMENTS.
C     NCHAR        THIS INTEGER VARIABLE CONTAINS THE NUMBER OF
C                  CHARACTERISTICS
C     TOTAL        THIS REAL VARIABLE RETURNS THE FLOWRATE OF THE
C                  FLOWSTREAM.
C
C     COMMON       VARIABLES USED
C     /WORK/       IBTU
C
C     VARIABLE     DESCRIPTION
C     NS           THIS INTEGER VARIABLE CONTAINS THE ROW NUMBER IN
C                  THE SUMMARY ARRAY OF THE TOTAL FLOW INFORMATION. IT
C                  IS THE ROW JUST BELOW THE ROW FOR THE LAST DETAILED
C                  SIZE INCREMENT.
C     I            THIS DUMMY COUNTER CONTROLS THE MOVEMENT THROUGH
C                  THE SIZE INCREMENTS.
C     J            THIS DUMMY COUNTER CONTROLS THE MOVEMENT THROUGH
C                  THE SPECIFIC GRAVITY INCREMENTS.
C     K            THIS DUMMY COUNTER CONTROLS THE MOVEMENT THROUGH
C                  THE ASH, SULFUR, AND BTU CATEGORIES.
C     EPSILN       THIS REAL VARIABLE CONTAINS THE EPSILON VALUE FOR
C                  ZERO.
C
      COMMON/WORK/NEXT,LAST,ICURVE,IUNIT,IDATA,ITITLE,ICHAR,IFLW,IASH,
     1 ISUL,IBTU
      DIMENSION FLOW(NSIZE,NGRAV,NCHAR),SUM(24,NCHAR)
      DATA EPSILN/1.E-8/
C
C     STEP 1       COMPUTE THE ROW NUMBER IN THE SUMMARY ARRAY OF THE
C                  ROW TO CONTAIN THE INFORMATION FOR THE TOTAL FLOW
C                  STREAM. THIS ROW IS JUST BELOW THE ROW FOR THE
```

FIGURE II-4-6.1. FORTRAN LISTING FOR SUBROUTINE SDSIZE

```
C                    FINAL SIZE INCREMENT.
C
      NS=NSIZE+1
C
C     STEP 2         COMPUTE THE SUMMARY ARRAY.
C
      DO 15 K=1,NCHAR
      SUM(NS,K)=0.0
      DO 15 I=1,NSIZE
      SUM(I,K)=0.0
      DO 10 J=1,NGRAV
      SUM(I,K)=SUM(I,K)+FLOW(I,J,K)
   10 CONTINUE
      SUM(NS,K)=SUM(NS,K)+SUM(I,K)
   15 CONTINUE
      TOTAL=SUM(NS,1)
      DO 25 I=1,NS
      IF(SUM(I,1).LT.EPSILN)GO TO 25
      DO 20 K=2,NCHAR
      SUM(I,K)=SUM(I,K)/SUM(I,1)
   20 CONTINUE
      SUM(I,1)=SUM(I,1)/TOTAL
   25 CONTINUE
C
C     STEP 3         MULTIPLY THE RATIOS IN THE SUMMARY ARRAY TIMES 100
C                    SO THAT THEY REFLECT PERCENTAGES.
C
      DO 50 I=1,NS
      DO 50 K=1,NCHAR
      IF(K.EQ.IPTU) GOTO 50
      SUM(I,K)=SUM(I,K)*100.
   50 CONTINUE
C
C     STEP 4         END LOCAL PROCESSING
C
      RETURN
      END
```

FIGURE II-4-6.1. (Continued)

4.7 Subroutine DSTRB

Subroutine DSTRB distributes a feed between two output flowstreams according to a distribution curve. Figure II-4-7.1 gives the listing and basic documentation for this routine.

```
      SUBROUTINE DSTRB(FEED,UPPER,LOWER,DISTRB,INCR,NSIZE,NGRAV,NCHAR)
C
C     TITLE(DISTRIBUTE THE FEED BETWEEN THE UPPER AND LOWER OUTPUT
C     FLOWSTREAMS)
C
C     SUBROUTINE DSTRB SPLITS THE FEED FLOWSTREAM INTO THE UPPER AND
C     LOWER OUTPUT FLOWSTREAMS ACCORDING TO A DISTRIBUTION CURVE.
C
C     PARAMETER    DESCRIPTION
C     FEED         THIS 3-DIMENSIONAL REAL ARRAY CONTAINS THE VALUES FOR
C                  THE FEED STREAM
C     UPPER        THIS 3-DIMENSIONAL REAL ARRAY RETURNS THE VALUES
C                  FOR THE UPPER OUTPUT FLOW.
C     LOWER        THS 3-DIMENSIONAL REAL ARRAY RETURNS THE VALUES FOR
C                  THE LOWER OUTPUT FLOW.
C     DISTRB       THIS 1-DIMENSIONAL REAL ARRAY CONTAINS THE
C                  DISTRIBUTION CURVE VALUES.
C     INCR         THIS 1-DIMENSIONAL INTEGER ARRAY INDICATES THE
C                  SIZE, SPECIFIC GRAVITY AND CHARACTERISTIC INCREMENTS
C                  IN THE DISTRIBUTION CURVE.
C     NSIZE        THIS INTEGER VARIABLE SPECIFIES THE NUMBER OF SIZE
C                  FRACTIONS CLASSIFYING THE FLOWSTREAM.
C     NGRAV        THIS INTEGER VARIABLE SPECIFIES THE NUMBER OF
C                  SPECIFIC GRAVITY FRACTIONS CLASSIFYING THE FLOWSTREAM
C     NCHAR        THIS INTEGER VARIABLE SPECIFIES THE NUMBER OF
C                  CHARACTERISTICS CLASSIFYING THE FLOWSTREAM.
C
C     VARIABLE     DESCRIPTION
C     I
C                  THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL
C                  MOVEMENT THROUGH THE SIZE FRACTIONS.
C     J            THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL
C                  MOVEMENT THROUGH THE SPECIFIC GRAVITY FRACTIONS.
C     K            THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL
C                  MOVEMENT THROUGH THE CHARACTERISTICS.
C     II           THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL
C                  MOVEMENT THROUGH THE SIZE FRACTIONS OF THE DISTRIB-
C                  UTION CURVE.
C     JJ           THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL
C                  MOVEMENT THROUGH THE SPECIFIC GRAVITY FRACTIONS OF
C                  THE DISTRIBUTION CURVE.
C     KK           THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL
C                  MOVEMENT THROUGH THE CHARACTERISTICS OF THE
C                  DISTRIBUTION CURVE.
C
      DIMENSION DISTRB(1),INCR(3)
      DIMENSION FEED(NSIZE,NGRAV,NCHAR),UPPER(NSIZE,NGRAV,NCHAR),
     1 LOWER(NSIZE,NGRAV,NCHAR)
      REAL LOWER
C
```

FIGURE II-4-7.1. LISTING OF SUBROUTINE DSTRB

```
C     STEP 1          INITIALIZE AT ONE THE DUMMY COUNTER USED TO CONTROL
C                     MOVEMENT THROUGH THE CHARACTERISTICS.
C
      DO 30 K=1,NCHAR
C
C     STEP 2          SET THE DUMMY VARIABLE CONTROLING THE CURRENT
C                     CHARACTERISTIC OF THE DISTRIBUTION CURVE.
C
      KK=K
      IF(INCR(3).EQ.0) KK=1
C
C     STEP 3          INITIALIZE AT ONE THE DUMMY COUNTER USED TO CONTROL
C                     MOVEMENT THROUGH THE SPECIFIC GRAVITY FRACTIONS.
C
      DO 20 J=1,NGRAV
C
C     STEP 4          SET THE DUMMY VARIABLE CONTROLING THE CURRENT
C                     SPECIFIC GRAVITY FRACTION OF THE DISTRIBUTION
C                     CURVE.
C
      JJ=J+(KK-1)*INCR(2)
      IF(INCR(2).EQ.0) JJ=KK
C
C     STEP 5          INITIALIZE AT ONE THE DUMMY COUNTER USED TO CONTROL
C                     MOVEMENT THROUGH THE SIZE FRACTIONS.
C
      DO 10 I=1,NSIZE
C
C     STEP 6          SET THE DUMMY VARIABLE CONTROLING THE CURRENT SIZE
C                     FRACTION OF THE DISTRIBUTION CURVE.
C
      II=I+(JJ-1)*INCR(1)
      IF(INCR(1).EQ.0) II=JJ
C
C     STEP 7          DISTRIBUTE THE FEED BETWEEN THE TWO OUTPUT STREAMS
C
      UPPER(I,J,K)=FEED(I,J,K)*DISTRB(II)
      LOWER(I,J,K)=FEED(I,J,K)-UPPER(I,J,K)
C
C     STEP 8          INCREMENT BY ONE THE DUMMY COUNTER CONTROLLING
C                     MOVEMENT THROUGH THE SIZE FRACTIONS AND IF ALL HAVE
C                     NOT BEEN PROCESSED LOOP BACK TO STEP 6.
10    CONTINUE
C     STEP 9          INCREMENT BY ONE THE DUMMY COUNTER CONTROLLING
C                     MOVEMENT THROUGH THE SPECIFIC GRAVITY FRACTIONS AND
C                     IF ALL HAVE NOT BEEN PROCESSED LOOP BACK TO STEP 4
20    CONTINUE
C     STEP 10         INCREMENT BY ONE THE DUMMY COUNTER CONTROLLING
C                     MOVEMENT THROUGH THE CHARACTERISTICS AND IF ALL
```

FIGURE II-4-7.1. (Continued)

```
C                    HAVE NOT BEEN PROCESSED LOOP BACK TO STEP 2.
30    CONTINUE
C     STEP 11        END LOCAL PROCESSING
      RETURN
      END
```

FIGURE II-4-7.1. (Continued)

4.8 Subroutine ECON

Subroutine ECON computes the economic data for the plant. Figure II-4-8.1 gives the listing and basic documentation for this routine.

```
      SUBROUTINE ECON
C
C     TITLE(COMPUTE ECONOMIC DATA)
C
C     SUBROUTINE ECON COMPUTES AN ECONOMIC SUMMARY FOR THE
C     OUTPUT FLOWSTREAMS.
C
C     COMMON          VARIABLES USED
C     /IOCNT/         CARCON, JOB
C     /CNTVAR/        KF, NCHAR, NFLOWS, S
C     //              WS, SG
C     /WORK/          IFLW, NEXT
C     /CODERC/        ICOL, LCOL, LINE
C     /CHRSET/        LETTER
C     /FILES/         READER
C     /BLK1/          CFD
C
C     VARIABLE        DESCRIPTION
C     LD1-LD13        THESE 1-DIMENSIONAL INTEGER ARRAYS CONTAIN INTERNAL
C                     DISPLAY CODES FOR OUTPUT PURPOSES.
C     PC              THIS REAL VARIABLE CONTAINS THE PRODUCTION CONSTANT.
C     IPC             THIS INTEGER VARIABLE CONTAINS THE PRODUCTION
C                     SPECIFICATION TYPE CODE.
C     IOP             THIS INTEGER VARIABLE CONTAINS THE SALEABLE OUTPUT
C                     FLOW CODE.
C     ICS             THIS INTEGER VARIABLE CONTAINS THE SALEABLE
C                     CHARACTERISTIC CODE.
C     HEATCV          THIS REAL VARIABLE IS THE BTU/LB TO HEAT UNIT
C                     CONVERSION FACTOR
C     NCONS           THIS INTEGER VARIABLE CONTAINS THE LENGTH OF
C                     CONSTRUCTION PERIOD.
C     NOPER           THIS INTEGER VARIABLE CONTAINS THE LENGTH OF THE
C                     OPERATIONAL PERIOD.
C     NTAB            THIS INTEGER VARIABLE CONTAINS THE NUMBER OF TABLE
C                     FUNCTIONS
C     NSCEN           THIS INTEGER VARIABLE CONTAINS THE NUMBER OF COST
C                     SCENARIOS.
C     F               THIS INTEGER VARIABLE IS USED AS A DO-LOOP COUNTER
C                     MOVING THROUGH THE FLOWS.
C     I               THIS INTEGER VARIABLE IS USED AS A DO-LOOP COUNTER.
C     FLOWP           THIS 1-DIMENSIONAL REAL ARRAY CONTAINS THE
C                     COMPOSITE CHARACTERISTICS OF PRODUCTION.
C     SINC            THIS 1-DIMENSIONAL REAL ARRAY CONTAINS THE HOLLERITH
C                     CHARACTERS C, M, R.
C     IBLNK           THIS INTEGER VARIABLE CONTAINS A DATA VALUE.
C     JF              THIS INTEGER VARIABLE IS A SUMMARY FLOWSTREAM POINTER
C     JF1             THIS INTEGER VARIABLE IS A SUMMARY FLOWSTREAM POINTER
C     IDCOST          THIS 1-DIMENSIONAL INTEGER ARRAY CONTAINS THE
C                     INTERNAL DISPLAY CODES FOR THE COST UNITS
C                     DESCRIPTION
C     IDPROD          THIS 1-DIMENSIONAL INTEGER ARRAY CONTAINS THE
C                     INTERNAL DISPLAY CODES FOR THE PRODUCTION UNITS
C                     DESCRIPTIONS
C     IDSCEN          THIS 1-DIMENSIONAL INTEGER ARRAY CONTAINS THE
C                     INTERNAL DISPLAY CODES FOR THE SCENARIO DESCRIPTIONS
C     AHOUR           THIS REAL VARIABLE CONTAINS THE NUMBER OF HOURS OF
```

FIGURE II-4-8.1. LISTING OF SUBROUTINE ECON

```
C                  OPERATION ANNUALLY.
C      PROD        THIS REAL VARIABLE CONTAINS THE AMOUNT OF MATERIAL
C                  IN THE ANNUAL PRODUCTION
C      NCELL       THIS INTEGER VARIABLE CONTAINS THE NUMBER OF CELLS
C                  USED IN DESCRIPTORS
C      ITAB        THIS INTEGER VARIABLE IS USED AS A POINTER FOR THE
C                  TABLE FUNCTION SPECIFICATIONS
C      LTAB        THIS INTEGER VARIABLE IS USED AS A POINTER FOR THE
C                  TABLE FUNCTION SPECIFICATIONS
C      ICE         THIS INTEGER VARIABLE IS USED AS A POINTER FOR THE
C                  COST ELEMENT INFORMATION
C      LCE         THIS INTEGER VARIABLE IS USED AS A POINTER FOR THE
C                  COST ELEMENT INFORMATION
C      ICELL       THIS INTEGER VARIABLE IS USED AS A POINTER IN
C                  WORKING STORAGE
C      LCELL       THIS INTEGER VARIABLE IS USED AS A POINTER IN
C                  WORKING STORAGE
C      IYR         THIS INTEGER VARIABLE IS THE FIRST YEAR OF THE
C                  OPERATIONAL PERIOD
C      LYR         THIS INTEGER VARIABLE IS THE LAST YEAR OF THE
C                  OPERATIONAL PERIOD
C      IPROF       THIS INTEGER VARIABLE CONTAINS THE PROFITABILITY
C                  MEASURE TYPE CODE
C      IDEP        THIS INTEGER VARIABLE CONTAINS THE DEPRECIATION
C                  TYPE CODE
C      IPROD       THIS INTEGER VARIABLE CONTAINS THE PRODUCTION
C                  ASSUMPTION CODE
C      NDEL        THIS INTEGER VARIABLE CONTAINS THE NUMBER OF COST
C                  ELEMENTS TO BE CHANGED
C      RP          THIS REAL VARIABLE CONTAINS THE RATE OF PROFIT
C                  DESIRED
C      DF          THIS REAL VARIABLE CONTAINS THE FRACTION OF INVEST-
C                  MENT TO BE BORROWED
C      RI          THIS REAL VARIABLE CONTAINS THE INTEREST RATE FOR
C                  THE DEBT
C      TR          THIS REAL VARIABLE CONTAINS THE INCOME TAX RATE
C      IBASE       THIS INTEGER VARIABLE CONTAINS THE BASE ADDRESS
C                  OF THE CURRENT COST ELEMENT
C      IARG,ARG    THIS TYPELESS VARIABLE CONTAINS THE COST
C                  CALCULATION ARGUMENT TYPE CODE
C      VAL,IVAL    THIS TYPELESS VARIABLE CONTAINS THE COST ELEMENT
C                  VALUE
C      IARG2       THIS INTEGER VARIABLE IS USED FOR TEMPORARY STORAGE
C      AP          THIS REAL ARRAY CONTAINS THE ANNUAL PRODUCTION
C      CD          THIS REAL ARRAY CONTAINS THE DEPRECIABLE CAPITAL
C                  INVESTMENT
C      CN          THIS REAL ARRAY CONTAINS THE NONDEPRECIABLE OPERATING
C                  INVESTMENT
C      OE          THIS REAL ARRAY CONTAINS THE OPERATING EXPENSES
C      DE          THIS REAL ARRAY CONTAINS THE DEPRECIATION
C      TA          THIS REAL ARRAY CONTAINS THE TAX ALLOWANCE
C      NY          THIS INTEGER VARIABLE CONTAINS THE NUMBER OF YEARS
C                  OF COMBINED CONSTRUCTION AND OPERATION
C      NPOINT      THIS INTEGER VARIABLE IS USED AS A POINTER FOR
C                  WORKING STORAGE
C      IY          THIS INTEGER VARIABLE CONTAINS THE INITIAL TIME VALUE
```

FIGURE II-4-8.1. (Continued)

```
C     LY              THIS INTEGER VARIABLE CONTAINS THE FINAL TIME VALUE
C     IROW            THIS INTEGER VARIABLE IS USED AS A ROW POINTER FOR
C                     OUTPUT
C     LROW            THIS INTEGER VARIABLE IS USED AS A ROW POINTER FOR
C                     OUTPUT
C     LINES           THIS INTEGER VARIABLE CONTAINS THE NUMBER OF LINES
C                     PRINTED ON THE CURRENT PAGE
C     LEVEL           THIS INTEGER ARRAY CONTAINS THE ADDRESSES IN WORKING
C                     STORAGE OF THE COST VALUES
C     J               THIS INTEGER VARIABLE IS USED AS A DO-LOOP COUNTER
C     IX              THIS INTEGER VARIABLE IS USED AS A POINTER FOR
C                     WORKING STORAGE
C     IT              THIS INTEGER VARIABLE IS USED AS A POINTER FOR
C                     WORKING STORAGE
C     ISC             THIS INTEGER VARIABLE IS USED AS A DO-LOOP COUNTER
C                     OVER THE SCENARIOS
C     IOPER           THIS INTEGER VARIABLE CONTAINS THE FIRST YEAR OF
C                     OPERATIONS
C     IL              THIS INTEGER VARIABLE IS USED AS A POINTER FOR
C                     WORKING STORAGE
C     ID
C     ICALC           THIS INTEGER VARIABLE CONTAINS THE COST CALCUL-
C                     ATION ARGUMENT TYPE CODE
C     IC              THIS INTEGER VARIABLE IS USED FOR TEMPORARY STORAGE
C
      COMMON/CHRSET/LETTER(47)
      COMMON/FILES/READER,PRINTR,OFFLIN
      COMMON/WORK/NEXT,LAST,ICURVE,IUNIT,IDATA,ITITLE,ICHAR,IFLW
      COMMON/CODERC/ICOL,LCOL,LINE(150)
      COMMON/CNTVAR/KF(2,100),SIZE(23),GBOUND(21),GRAV(20),WFLOW(100)
     1             ,S(100),D1(60),D2(60),D3(60),L(60)
     2                          ,NSIZE,NGRAV,IOUT,NUNITS,NFLOWS,NCHAR
      COMMON/IOCNT/JOB,CARCON
      COMMON/BLK1/CFD(50,6),CFT(50,6)
      COMMON WS(6000)
      COMMON/LOCALV/PC,IPC,IOP,ICS,HEATCV,NCONS,NOPER,NTAB,NSCEN
     1             ,FLOWP(5),AFEED,PROD,IDCOST(5),IDPROD(5),ITAB,ID
     2             ,NCELL,LTAB,LCELL,ICELL,ICE,LCE,IBASE,IYR,LYR,IPROF
     3             ,IDEP,IPROD,NDEL,RP,DF,RI,TR,LEVEL(10),IDSCEN(11)
      DIMENSION SINC(3)
      DIMENSION AP(50),CD(50),CN(50),OE(50),DE(50),TA(50)
      INTEGER READER,PRINTR,OFFLIN,WS
      INTEGER F
      INTEGER CARCON
      DIMENSION LD1(8),LD2(8),LD3(8)
      DIMENSION SG(6000)
      EQUIVALENCE (WS(1),SG(1))
      EQUIVALENCE (AP(1),CFD(1,1)),(CD(1),CFD(1,2)),(CN(1),CFD(1,3))
      EQUIVALENCE (OE(1),CFD(1,4)),(DE(1),CFD(1,5)),(TA(1),CFD(1,6))
      EQUIVALENCE (VAL,IVAL),(IARG,ARG)
      DATA SINC(1),SINC(2),SINC(3)/1HC,1HM,1HR/
      DATA IBLNK/-999999/
      DATA LD1(1),LD1(2),LD1(3),LD1(4)/34,203765001,236901318,755823928/
      DATA LD1(5),LD1(6),LD1(7)/756334925,87640918,20009299/
      DATA LD1(8)/11822034/
      DATA LD2(1),LD2(2),LD2(3),LD2(4)/33,273216133,55759041,321048972/
```

FIGURE II-4-8.1. (Continued)

```
      DATA LD2(5),LD2(6),LD2(7)/257872133,319627856,337965997/
      DATA LD2(8)/25554/
      DATA LD3(1),LD3(2),LD3(3),LD3(4)/33,69546191,356073796,755766472/
      DATA LD3(5),LD3(6),LD3(7)/756597719,755294284,424449261/
      DATA LD3(8)/25554/
C
C     STEP 1        READ THE MAIN COST COMPUTATION CONTROL CARD.
C
      DO 6 I=1,150
    6 LINE(I)=LETTER(45)
    1 FORMAT(1F10.0,3I1,1F10.0,2I5,1I3,1I4,40A1)
      READ(READER,1)PC,IPC,IOP,ICS,HEATCV,NCONS,NOPER,NTAB,NSCEN
     1              ,(LINE(I),I=1,40)
C
C     STEP 2        COMPUTE THE COMPOSITE CHARACTERISTICS OF THE SALABLE
C                   OUTPUT FROM THE CONFIGURATION.
C
      DO 10 I=1,5
      FLOWP(I)=0.0
   10 CONTINUE
      DO 30 F=1,NFLOWS
      IF(KF(2,F).NE.0)GO TO 30
      IF(IOP.EQ.0)GO TO 20
      DO 15 I=1,IOP
      IF(S(F).EQ.SINC(I))GO TO 20
   15 CONTINUE
      GO TO 30
   20 JF=IFLW+NCHAR *(F-1)
      FLOWP(1)=FLOWP(1)+SG(JF+1)
      DO 25 I=2,5
      JF1=JF+I
      FLOWP(I)=FLOWP(I)+SG(JF1)
   25 CONTINUE
   30 CONTINUE
C
C     STEP 3        COMPUTE THE NUMBER OF HOURS OF OPERATION OF THE
C                   PROJECT ANNUALLY, AND THE AMOUNT OF MATERIAL IN THE
C                   PRODUCTION.
C
      IF(IPC.EQ.0)GO TO 55
      AHOUR=PC/SG(IFLW+1)
      GO TO 75
   55 AHOUR=PC/FLOWP(1)
   75 PROD=AHOUR*FLOWP(1)
C
C     STEP 4        STORE THE COST UNITS AND PRODUCTION UNITS DESCRIP-
C                   TIONS USIND SUBROUTINE  R D D E S C .
C
      ICOL=1
      LCOL=20
      CALL RDDESC(IDCOST,NCELL)
      ICOL=21
      LCOL=40
      CALL RDDESC(IDPROD,NCELL)
C
C     STEP 5        READ THE TABLE FUNCTION SPECIFICATIONS AND STORE
```

FIGURE II-4-8.1. (Continued)

```
C                    THEM IN WORKING STORAGE.
C
      IF(NTAB.EQ.0)GO TO 85
      ITAB=NEXT+1
      DO 80 IT=1,NTAB
      READ(READER,2)ID,NCELL
    2 FORMAT(2I5)
      WS(NEXT)=ID
      NEXT=NEXT+1
      IF(IT.GT.1)WS(LTAB)=NEXT
      LTAB=NEXT
      WS(NEXT)=0
      NEXT=NEXT+1
      WS(NEXT)=NCELL
      NEXT=NEXT+1
      LCELL=NEXT+NCELL-1
      READ(READER,3)(WS(I),I=NEXT,LCELL)
    3 FORMAT(8F10.0)
      NEXT=LCELL+1
      LCELL=LCELL+NCELL
      READ(READER,3)(WS(I),I=NEXT,LCELL)
      NEXT=LCELL+1
   80 CONTINUE
C
C     STEP 6        READ AND STORE THE BASIC COST ELEMENT INFORMATION.
C
   85 ICE=NEXT+1
      LCE=0
   90 IBASE=NEXT+1
      WS(IBASE)=0
      ICELL=IBASE+6
      LCELL=IBASE+11
    4 FORMAT(1I3,2I1,6A1,1I1,1I3,1I1,1I3,1X,2F10.0,40A1)
      READ(READER,4)WS(IBASE-1),WS(IBASE+2),WS(IBASE+3),(LINE(I),I=1,6)
     1              ,(WS(I),I=ICELL,LCELL),(LINE(I),I=7,46)
      IF(WS(IBASE-1).EQ.-1)GO TO 100
      NEXT=NEXT+13
      IF(LCE.GT.0)WS(LCE)=IBASE
      LCE=IBASE
      IF(WS(IBASE+3).EQ.0)GO TO 97
      ICOL=1
      LCOL=3
      CALL RDVAL(IYR,IBLNK)
      IF(IYR.EQ.IBLNK)GO TO 91
      ICOL=4
      LCOL=6
      CALL RDVAL(LYR,IYR)
      GO TO 96
   91 IF(WS(IBASE+3)-1)93,94,92
   92 IF(WS(IBASE+3)-4)95,94,94
   93 IYR=0
      LYR=0
      GO TO 96
   94 IYR=1
      LYR=NOPER
      GO TO 96
```

FIGURE II-4-8.1. (Continued)

```
   95 IYR=-NCONS+1
      LYR=IYR
   96 WS(IBASE+4)=IYR+NCONS
      WS(IBASE+5)=LYR+NCONS
   97 ICOL=7
      LCOL=46
      CALL RDDESC(WS(NEXT),NCELL)
      NEXT=NEXT+NCELL
      GO TO 90
C
C     STEP 7         BEGIN PROCESSING THE INDIVIDUAL COST SCENARIOS.
C                    BEGIN BY INITIALIZING AT ONE THE DUMMY COUNTER USED
C                    TO CONTROL THE MOVEMENT THROUGH THE COST COMPUTATION
C                    SCENARIOS.
C
  100 DO 600 ISC=1,NSCEN
C
C     STEP 8         READ THE CURRENT COST SCENARIO CONTROL CARD.
C
    5 FORMAT(3I1,1I2,4F5.0,5X,50A1)
      READ(READER,5)IPROF,IDEP,IPROD,NDEL,RP,DF,RI,TR,(LINE(I),I=1,50)
      ICOL=1
      LCOL=50
      CALL RDDESC(IDSCEN,NCELL)
C
C     STEP 9         READ THE COST ELEMENT CHANGES
C
      IF(NDEL.EQ.0)GO TO 130
      DO 125 IC=1,NDEL
      ID=IC
  125 CONTINUE
C
C     STEP 10        INITIALIZE ALL COST ELEMENT VALUES AT ZERO.
C
  130 IBASE=ICE
      VAL=0.0
  135 WS(IBASE+1)=IVAL
      IBASE=WS(IBASE)
      IF(IBASE.GT.0)GO TO 135
      DO 136 I=1,10
      LEVEL(I)=0
  136 CONTINUE
C
C     STEP 11        BEGIN THE COMPUTATION OF THE COST ELEMENT VALUES.
C                    FIRST SET THE BASE ADDRESS OF THE CURRENT COST
C                    ELEMENT EQUAL TO THE BASE ADDRESS OF THE FIRST
C                    ELEMENT.
C
      IBASE=ICE
C
C     STEP 12        OBTAIN THE VALUE OF THE ARGUMENT FOR THE COST ELEMENT
C                    COMPUTATION FOR THE CURRENT ELEMENT.
C
  139 IARG=WS(IBASE+6)
      IF(IARG-1)140,145,160
  140 IARG=WS(IBASE+10)
```

FIGURE II-4-8.1. (Continued)

```
      GO TO 200
  145 ICELL=ICE
  150 IF(WS(IBASE+7).EQ.WS(ICELL-1))GO TO 155
      ICELL=WS(ICELL)
      IF(ICELL) 150,995,150
  155 IARG=WS(ICELL+1)
      GO TO 200
  160 F=WS(IBASE+7)
      IF(F.EQ.0)GO TO 180
      IF(F.GT.NFLOWS)GO TO 996
      JF=IFLW+NCHAR *(F-1)
      IF(IARG-3)165,170,175
  165 ARG=SG(JF+1)*AHOUR
      GO TO 200
  170 ARG=AHOUR*SG(JF+5)*HEATCV
      GO TO 200
175   IARG2=IARG-2+JF
      ARG=AHOUR*SG(IARG2)
      GO TO 200
  180 IF(IARG-3)185,190,195
  185 ARG=FLOWP(1)*AHOUR
      GO TO 200
  190 ARG=AHOUR*FLOWP(5)*HEATCV
      GO TO 200
  195 ARG=AHOUR*FLOWP(IARG-2)
C
C     STEP 13       COMPUTE THE COST ELEMENT VALUE
C
  200 ICALC=WS(IBASE+8)
      IF(ICALC-1)205,210,201
  201 IF(ICALC-3)215,220,220
  205 VAL=ARG
      GO TO 250
  210 IVAL=WS(IBASE+11)
      VAL=VAL*ARG
      GO TO 250
  215 IVAL=WS(IBASE+11)
      VAL=ARG**VAL
      GO TO 250
  220 IF(NTAB.EQ.0)GO TO 997
      IT=ITAB
  225 IF(WS(IT-1).EQ.WS(IBASE+9))GO TO 230
      IT=WS(IT)
      IF(IT)225,998,225
  230 NPOINT=WS(IT+2)
      IX=IT+3
      IY=IX+NPOINT
      IF(WS(IT+1).EQ.1)GO TO 235
      CALL INTS(WS(IX),WS(IY),ARG,VAL,NPOINT)
      GO TO 250
  235 VAL=YINTRP(WS(IX),WS(IY),ARG,NPOINT)
C
C     STEP 14       ADD THE COMPUTED COST VALUE INTO ALL THE APPROPRIATE
C                   COST VALUE CELLS, AND UPDATE THE LEVEL ADDRESSES.
C
  250 IL=WS(IBASE+2)+1
```

FIGURE II-4-8.1. (Continued)

```
      LEVEL(IL)=IBASE
      DO 255 I=1,IL
      ICELL=LEVEL(I)
      IF(ICELL.EQ.0) GOTO 255
      IARG=WS(ICELL+1)
      ARG=ARG+VAL
      WS(ICELL+1)=IARG
  255 CONTINUE
C
C     STEP 15       OBTAIN THE BASE ADDRESS OF THE NEXT COST ELEMENT,
C                   AND IF IT IS NOT ZERO, LOOP BACK TO STEP 12.
C
      IBASE=WS(IBASE)
      IF(IBASE.GT.0)GO TO 139
C
C     STEP 16       PRINT THE BASIC TABLE FOR THE COST VALUES.
C
      IBASE=ICE
      LINES=100
  260 IF(WS(IBASE+12).EQ.0)GO TO 286
      IF(LINES.LT.25)GO TO 265
      LINES=0
      CALL PRHEAD
      JOB=0
      CARCON=LETTER(27)
      ICOL=1
      CALL PRDESC(LD1)
      ICOL=ICOL+1
      LINE(ICOL)=LETTER(45)
      CALL PRDESC(IDSCEN)
      CALL PRLINE
      CARCON=LETTER(45)
      CALL PRLINE
  265 LINES=LINES+1
  270 IF(ICOL.GT.WS(IBASE+2))GO TO 275
      ICOL=ICOL+1
      LINE(ICOL)=LETTER(45)
      GO TO 270
  275 CALL PRDESC(WS(IBASE+12))
  280 IF(ICOL.GE.50)GO TO 285
      ICOL=ICOL+1
      LINE(ICOL)=LETTER(47)
      GO TO 280
  285 CONTINUE
      CARCON=LETTER(27)
      CALL CNVTIE(WS(IBASE+1),16,2)
      CALL PRLINE
  286 IBASE=WS(IBASE)
      IF(IBASE.GT.0)GO TO 260
C
C     STEP 17       INITIALIZE THE PROJECT CASH FLOW DESCRIPTION AT ZERO.
C
      NY=NCONS+NOPER
      DO 290 I=1,NY
      DO 290 J=1,6
      CFD(I,J)=0.0
```

FIGURE II-4-8.1. (Continued)

```
  290 CONTINUE
C
C     STEP 18      ADD THE COST ELEMENT VALUES INTO THE CASH FLOW
C                  DESCRIPTION
C
      IBASE=ICE
  295 IC=WS(IBASE+3)
      IF(IC.EQ.0)GO TO 305
      IY=WS(IBASE+4)
      LY=WS(IBASE+5)
      IVAL=WS(IBASE+1)
      DO 300 I=IY,LY
      CFD(I,IC)=CFD(I,IC)+VAL
  300 CONTINUE
  305 IBASE=WS(IBASE)
      IF(IBASE.GT.0)GO TO 295
C
C     STEP 19      COMPUTE DEPRECIATION
C
      IOPER=NCONS+1
      IF(IDEP.EQ.0)GO TO 335
      TCD=0.0
      DO 310 I=1,NCONS
      TCD=TCD+CD(I)
  310 CONTINUE
      DO 315 I=IOPER,NY
      DE(I)=TCD/FLOAT(NY-NCONS)
  315 CONTINUE
      DO 325 J=IOPER,NY
      IF(J.EQ.NY)GO TO 325
      IF(CD(J).EQ.0.0)GO TO 325
      IJ=J+1
      DO 320 I=IJ,NY
      DE(I)=DE(I)+CD(J)/FLOAT(NY-J)
  320 CONTINUE
  325 CONTINUE
C
C     STEP 20      ALLOCATE PRODUCTION
C
  335 IF(IPROD)360,340,350
  340 DO 345 I=IOPER,NY
      AP(I)=AP(I)+PROD
  345 CONTINUE
      GO TO 360
  350 DO 355 I=IOPER,NY
      AP(I)=AP(I)*(OE(I)/OE(IPROD))
  355 CONTINUE
C
C     STEP 21      DISPLAY THE PROJECT CASH FLOW DESCRIPTION.
C
  360 LROW=0
      IY=-NCONS
  365 IROW=LROW+1
      LROW=LROW+23
      IF(LROW.GT.NY)LROW=NY
      CALL PRHEAD
```

FIGURE II-4-8.1. (Continued)

```
      JOB=0
      CARCON=LETTER(27)
      ICOL=1
      CALL PRDESC(LO2)
      ICOL=ICCL+1
      CALL PRDESC(IDSCEN)
      CALL PRLINE
      CALL PRLINE
      WRITE(PRINTR,601)
  601 FORMAT(36X,11HDEPRECIABLE,8X,14HNONDEPRECIABLE,/,20X,6HANNUAL
     1       ,13X,7HCAPITAL,13X,9HOPERATING,51X,9HOTHER TAX,/,4X,4HYEAR
     2       ,10X,10HPRODUCTION,10X,10HINVESTMENT,10X,10HINVESTMENT,12X
     3       ,8HEXPENSES,8X,12HDEPRECIATION,10X,10HALLOWANCES)
      DO 370 I=IROW,LROW
      IY=IY+1
      WRITE(PRINTR,602)IY,(CFD(I,J),J=1,6)
  602 FORMAT(1H0,2X,I4,6F20.2)
  370 CONTINUE
      IF(LROW.LT.NY)GO TO 365
      DO 375 I=1,NY
      DO 375 J=1,6
      CFT(I,J)=0.0
  375 CONTINUE
      CFT(1,5)=1.0/(1.0+RP)**(0-NCONS)
      DO 380 I=2,NY
      CFT(I,5)=CFT(I-1,5)/(1.0+RP)
  380 CONTINUE
      CNPV=0.0
      DO 385 I=1,NCONS
      CFT(I,1)=0.0-OE(I)-DE(I)-TA(I)
      IF(CFT(I,1).GT.0.0)CFT(I,2)=CFT(I,1)*TR
      CFT(I,3)=0.0-OE(I)-CFT(I,2)
      CFT(I,4)=CFT(I,3)-CD(I)-CN(I)
      CFT(I,6)=CFT(I,4)*CFT(I,5)
      CNPV=CNPV+CFT(I,6)
  385 CONTINUE
      AS=0.0
      SUMDF=0.0
      DO 390 I=IOPER,NY
      AS=AS+(OE(I)*(1.0-TR)-(DE(I)+TA(I))*TR+CD(I)+CN(I))*CFT(I,5)
      SUMDF=SUMDF+CFT(I,5)
  390 CONTINUE
      AS=(AS-CNPV)/((1.0-TR)*SUMDF)
  395 TNPV=CNPV
      DO 400 I=IOPER,NY
      CFT(I,1)=AS-OE(I)-DE(I)-TA(I)
      CFT(I,2)=0.0
      IF(CFT(I,1).GT.0.0)CFT(I,2)=CFT(I,1)*TR
      CFT(I,3)=AS-OE(I)-CFT(I,2)
      CFT(I,4)=CFT(I,3)-CN(I)-CD(I)
      CFT(I,6)=CFT(I,4)*CFT(I,5)
      TNPV=TNPV+CFT(I,6)
  400 CONTINUE
      IF(ABS(TNPV).LE.0.01)GO TO 405
      AS=AS-TNPV/((1.0-TR)*SUMDF)
      GO TO 395
```

FIGURE II-4-8.1. (Continued)

```
405 AS=AS/AP(NY)
    LROW=0
    IY=-NCONS
410 IROW=LROW+1
    LROW=LROW+23
    IF(LROW.GT.NY)LROW=NY
    CALL PRHEAD
    JOB=0
    CARCON=LETTER(27)
    ICOL=1
    CALL PRDESC(LD3)
    ICOL=ICOL+1
    CALL PRDESC(IDSCEN)
    CALL PRLINE
    CALL PRLINE
    WRITE(PRINTR,603)
603 FORMAT(59X,9HCASH FLOW,/,21X,7HTAXABLE,15X,5HTAXES,13X,5HAFTER
   1       ,16X,3HNET,15X,8HDISCOUNT,10X,10HDISCOUNTED,/,4X,4HYEAR
   2       ,14X,6HINCOME,16X,4HPAID,13X,5HTAXES,13X,9HCASH FLOW,13X
   3       ,6HFACTOR,12X,9HCASH FLOW)
    DO 415 I=IROW,LROW
    IY=IY+1
    WRITE(PRINTR,604)IY,(CFT(I,J),J=1,6)
604 FORMAT(1H0,2X,I4,4F20.2,F20.6,F20.4)
415 CONTINUE
    IF(LROW.LT.NY)GO TO 410
605 FORMAT(//,1X,19HUNIT SELLING PRICE ,1F10.4,16H RATE OF RETURN=
   1,1F6.4,17H INCOME TAX RATE=,1F6.4)
    WRITE(PRINTR,605)AS,RP,TR
600 CONTINUE
    GO TO 999
995 STOP 1
996 STOP 2
997 STOP 3
998 STOP 4
999 RETURN
    END
```

FIGURE II-4-8.1. (Continued)

5.0 THE UNIT OPERATIONS

By now just about everything that can be said about the unit operations has been said. This chapter presents the actual routines which simulate the unit operations. The actual code within these routines is self-evident and straightforward given all of the previous discussion. Other than the listings themselves and the basic documentation associated with them little additional information is provided in this chapter.

5.1 Subroutine BREAK2

Subroutine BREAK2 simulates breakage during the operation of a unit. Figure II-5-1.1 gives the listing and basic documentation for this routine.

```
      SUBROUTINE BREAK2(FEED,PROD,BR,SIZE,NSIZE,NGRAV,NCHAR,BRK,NBRK)
C
C     TITLE(SIMULATE BREAKING OF FLOW DURING UNIT OPERATION)
C
C
C     SUBROUTINE BREAK2 IS THE ROUTINE USED TO BREAK THE FLOW BEFORE IT
C     ENTERS A UNIT.  IT SIMULATES THE BREAKAGE THAT OCCURS DURING
C     OPERATION OF THE UNIT.
C
C     PARAMETER    DESCRIPTION
C     FEED         THIS 3-DIMENSIONAL REAL ARRAY CONTAINS THE DETAILED
C                  CHARACTERISTICS OF THE FLOW BEFORE BREAKAGE OCCURS.
C                  IT RETURNS THE DETAILS OF THE FLOW AFTER BREAKAGE.
C     PROD         THIS 3-DIMENSIONAL REAL ARRAY IS USED FOR SCRATCH
C                  STORAGE.
C     BR           THIS 2-DIMENSIONAL REAL ARRAY IS USED TO CONTAIN
C                  THE BREAKAGE MATRIX.
C     SIZE         THIS 1-DIMENSIONAL REAL ARRAY CONTAINS THE SIZE
C                  FRACTION DISTRIBUTION CLASSIFYING THE FLOWSTREAM.
C     NSIZE        THIS INTEGER VARIABLE CONTAINS THE NUMBER OF SIZE
C                  FRACTIONS CLASSIFYING THE FLOWSTREAM.
C     NGRAV        THIS INTEGER VARIABLE CONTAINS THE NUMBER OF SPECIFIC
C                  GRAVITY FRACTIONS CLASSIFYING THE FLOWSTREAM.
C     NCHAR        THIS INTEGER VARIABLE CONTAINS 1 PLUS THE NUMBER OF
C                  FLOWSTREAM CHARACTERISTICS.
C     BRK          THIS 1-DIMENSIONAL REAL ARRAY CONTAINS THE BREAKAGE
C                  SPECIFICATIONS FOR THE UNIT.
C     NBRK         THIS INTEGER VARIABLE CONTAINS THE NUMBER OF CELLS
C                  USED IN SPECIFYING THE BREAKAGE IN THE UNIT.
C
C     COMMON       VARIABLES USED
C     /BLK1/       SUMRY, ISCC
C
C     EXTERNALS    BRSIZE, CMPBRK, SDSIZE
C
C     VARIABLE     DESCRIPTION
C     I,J,K,L      THESE DUMMY INTEGER VARIABLES ARE USED TO CONTROL THE
C                  MOVEMENT THROUGH MISCELLANEOUS LOACL LOOPS.
C     STORE        THIS REAL VARIABLE IS USED AS A TEMPORARY STORAGE
C                  LOCATION.
C     BRG,IBRG     THIS TYPELESS, 1-DIMENSIONAL ARRAY IS USED TO CONTAIN
C                  THE BRIDGE SPECIFICATION ARRAY FOR MOVING FROM THE
C                  UNIT BREAKAGE SIZE SPECIFICATION TO THE FLOWSTREAM
C                  SPECIFICATION.
C     NBRG         THIS INTEGER VARIABLE CONTAINS THE NUMBER OF CELLS
C                  USED IN THE BRIDGE SPECIFICATION ARRAY.
C     NSB          THIS INTEGER VARIABLE CONTAINS THE NUMBER OF SIZE
C                  FRACTIONS SPECIFYING THE BREAKAGE IN THE UNIT.
C     IB           THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL
```

FIGURE II-5-1.1. LISTING FOR SUBROUTINE BREAK2

```
C                         THE MOVEMENT THROUGH THE BRIDGE SPECIFICATION ARRAY.
C      SF                 THIS 1-DIMENSIONAL REAL ARRAY CONTAINS THE SELECTION
C                         FACTORS FOR THE UNIT BREAKAGE SIZE SPECIFICATION.
C      SI                 THIS 1-DIMENSIONAL REAL ARRAY IS USED TO CONTAIN
C                         THE SELECTION FACTORS FOR THE FLOWSTREAM SIZE
C                         SPECIFICATION.
C      NB                 THIS INTEGER VARIABLE CONTAINS THE NUMBER OF
C                         FLOWSTREAM SIZE FRACTIONS TO WHICH A GIVEN UNIT
C                         BREAKAGE SIZE FRACTION CONTRIBUTES.
C      TOTAL              THIS REAL VARIABLE RETURNS THE TOTAL AMOUNT, BY
C                         WEIGHT, OF COAL IN THE BROKEN FLOW
C      NS                 THIS INTEGER VARIABLE CONTAINS 1 LESS THAN THE
C                         NUMBER OF SIZE FRACTIONS CLASSIFYING THE VARIABLES.
C      VAL,IVAL           THIS TYPELESS VARIABLE IS USED TO READ REAL VARIABLES
C                         FROM THE INTEGER ARRAY BRK.
C
       DIMENSION FEED(NSIZE,NGRAV,NCHAR),PROD(NSIZE,NGRAV,NCHAR)
      1          ,BR(NSIZE,NSIZE),SIZE(NSIZE),BRK(NBRK),IBRG(100)
       COMMON/LOCALV/BRG(100),NSB,NBRG,IB,SF(23),SI(23),NB,STOPE,
      1 TOTAL,LVFILL(43)
       COMMON/BLK1/ISFD,ISCC,ISRF,ISMX,SYLD(24),TYLD(24),EFFIC(24),
      1 BTUREC(24),ASHERR(24),SFLT(24),SINK(24),SMISPL(24),SNRGR(24),
      2 SGRAV(24),SPE(24),SIMP(24),SEA(24),SUMRY(24,40)
       INTEGER BRK
       EQUIVALENCE (BRG(1),IBRG(1)),(VAL,IVAL)
C
C      STEP 1             COMPUTE THE BREAKAGE MATRIX USING SUBROUTINE
C                         C M P B R K.
C
       CALL CMPBRK(BR,PROD, BRK(2),SIZE,NSIZE)
C
C      STEP 2             SET THE NUMBER OF SIZE FRACTIONS FOR UNIT BREAKAGE.
C
       NSB= BRK(1)
C
C      STEP 3             IF THE SELECTION FACTORS ARE THE SAME FOR ALL SIZE
C                         FRACTIONS, BRANCH TO STEP 10.
C
       IF(NSB.EQ.1) GOTO 40
C
C      STEP 4             USING SUBROUTINE  B R S I Z E  COMPUTE THE
C                         SPECIFICATIONS NEEDED TO MOVE FROM THE UNIT
C                         BREAKAGE SIZE DISTRIBUTION TO THE FLOWSTREAM SIZE
C                         DISTRIBUTION.
C
       CALL BRSIZE(SIZE,NSIZE,BRK(3),NSB,BRG,NBRG)
C
C      STEP 5             ZERO THE CONTENTS OF THE ARRAY TO CONTAIN THE
C                         SELECTION FACTORS FOR THE FLOWSTREAM SIZE
```

FIGURE II-5-1.1. (Continued)

```
C                     SPECIFICATION.
C
      DO 20 I=1,NSIZE
      SI(I)=0.0
   20 CONTINUE
C
C     STEP 6          SET THE CONTENTS OF THE ARRAY TO CONTAIN THE
C                     SELECTION FACTORS FOR THE BREAKAGE SIZE SPECIFICATION
C
      DO 25 I=1,NSB
      J=I+NSB+3
      IVAL=BRK(J)
      SF(I)=VAL
   25 CONTINUE
C
C     STEP 7          INITIALIZE THE VARIABLE NEEDED TO MOVE THROUGH
C                     THE BREAKAGE SIZE DISTRIBUTION.
C
      IB=0
C
C     STEP 8          MOVING THROUGH THE SIZE FRACTIONS COMPUTE THE
C                     SELECTION FACTORS FOR THE FLOWSTREAM SIZE
C                     DISTRIBUTION.
C
      DO 35 I=1,NSB
      IB=IB+1
      NB=IBRG(IB)
      IF(NB.EQ.0) GOTO 35
      DO 30 J=1,NB
      IB=IB+1
      K=IBRG(IB)
      IB=IB+1
      SI(K)=SI(K)+BRG(IB)*SF(I)
   30 CONTINUE
   35 CONTINUE
C
C     STEP 9          BRANCH TO STEP 11
C
      GOTO 50
C
C     STEP 10         SET THE ARRAY OF SELECTION FACTORS FOR THE CASE
C                     WHERE ALL SELECTION FACTORS ARE EQUAL.
C
   40 CONTINUE
      IVAL=BRK(5)
      DO 45 I=1,NSIZE
      SI(I)=VAL
   45 CONTINUE
C
```

FIGURE II-5-1.1. (Continued)

```
C     STEP 11       COMPUTE THE PRODUCT AFTER BREAKAGE.
C
   50 CONTINUE
      NS=NSIZE-1
      DO 60 I=1,NS
      DO 60 J=1,NGRAV
      DO 60 K=1,NCHAR
      STORE=0.0
      DO 55 L=1,I
      STORE=STORE+FEED(L,J,K)*BR(L,I)*SI(L)
   55 CONTINUE
      PROD(I,J,K)=FEED(I,J,K)*(1.-SI(I))+STORE
   60 CONTINUE
      DO 70 J=1,NGRAV
      DO 70 K=1,NCHAR
      STORE=0.0
      DO 65 L=1,NS
      STORE=STORE+FEED(L,J,K)*BR(L,NSIZE)*SI(L)
   65 CONTINUE
      PROD(NSIZE,J,K)=FEED(NSIZE,J,K)+STORE
   70 CONTINUE
      DO 75 I=1,NSIZE
      DO 75 J=1,NGRAV
      DO 75 K=1,NCHAR
      FEED(I,J,K)=PROD(I,J,K)
   75 CONTINUE
C
C     STEP 12       USING SUNROUTINE  S D S I Z E  COMPUTE THE SUMMARY
C                   DATA FOR THE BREAKAGE PRODUCT.
C
      CALL SDSIZE(FEED,SUMRY(1,ISCC),NSIZE,NGRAV,NCHAR,TOTAL)
      RETURN
      END
```

FIGURE II-5-1.1. (Continued)

5.2 Subroutine NOTPTH

A calculation required primarily by the screening unit operation and used also by the rotary breaking operation is of the proportion of material of a certain size that will not pass through an opening of a given size. This calculation was described in Volume I in the part on the screening unit operation. Subroutine NOTPTH performs it. Figure II-5-2.1 contains the listing and basic documentation.

```
      SUBROUTINE NOTPTH(SIZE,PR,HOLE,AA,NSIZE)
C
C     TITLE(PROPORTION NOT PASSING THROUGH HOLE)
C
C     SUBROUTINE NOTPTH COMPUTES THE PROPORTION OF MATERIAL CLASSIFIED
C     BY SIZE INCREMWNTS WHICH WILL NOT PASS THROUGH A HOLE OF A
C     SPECIFIED SIZE.
C
C     PARAMETER      DESCRIPTION
C     SIZE(1)        THIS REAL 1-DIMENSIONAL,MONOTONE-DECREASING, REAL
C                    ARRAY DEFINES THE SIZE INCREMENTS FOR THE MATERIAL
C                    TO BE PASSED THROUGH. THE I-TH CELL DEFINES THE
C                    UPPER BOUND FOR THE I-TH INCREMENT, WHILE THE I+1ST
C                    CELL DEFINES THE LOWER BOUND FOR THE I-TH INCREMENT.
C     PR(NSIZE)      THIS 1-DIMENSIONAL REAL ARRAY RETURNS THE PROPORTION
C                    OF THE MATERIAL BY SIZE INCREMENT THAT WILL NOT PASS
C                    THROUGH THE HOLE.
C     HOLE           THIS REAL VARIABLE CONTAINS THE SIZE OF THE HOLE.
C     AA             THIS REAL VARIABLE CONTAINS THE ADJUSTMENT FACTOR
C                    USED IN THE COMPUTATION OF THE PROPORTION. IT IS A
C                    FUNCTION OF THE TYPE OF UNIT AND HOLE SIZE.
C     NSIZE          THIS INTEGER VARIABLE DEFINES THE NUMBER OF SIZE
C                    INCREMENTS.
C
C     VARIABLE       DESCRIPTION
C     I              THIS DUMMY COUNTER CONTROLS THE MOVEMENT THROUGH
C                    THE SIZE INCREMENTS.
C     FRAC           THIS REAL VARIABLE CONTAINS THE PROPORTION OF THE
C                    MATERIAL IN A GIVEN SIZE INCREMENT WHICH IS LARGER
C                    THAN THE HOLE SIZE.
C     SMID           THIS REAL VARIABLE CONTAINS THE AVERAGE MATERIAL
C                    SIZE RELATIVE TO THE HOLE SIZE FOR A GIVEN SIZE
C                    INCREMENT.
C                    THIS REAL VARIABLE CONTAINS THE PROPORTION OF MAT-
C                    ERIAL IN A GIVEN SIZE INCREMENT WHICH IS SMALLER
C                    THAN THE HOLE SIZE WHICH WILL NOT PASS THROUGH.
C     SMAX           THIS REAL CONTAINS THE MAXIMUM OF THE CURRENT SIZE
C                    FRACTION.
C     C              THIS REAL VARIABLE IS USED FOR TEMPORARY STORAGE.
C
      DIMENSION SIZE(1),PR(NSIZE)
C
C     STEP 1         INITIALIZE AT ONE THE DUMMY COUNTER WHICH MOVES
C                    THROUGH THE SIZE INCREMENTS.
C
      DO 50 I=1,NSIZE
C
C     STEP 2         IF THE UPPER BOUND OF THE CURRENT SIZE INCREMENT
C                    DOES NOT EXCEED THE HOLE SIZE, BRANCH TO STEP 6.
```

FIGURE II-5-2.1. LISTING FOR SUBROUTINE NOTPTH

```
C
      IF(SIZE(I)-HOLE)25,25,10
C
C     STEP 3          IF THE LOWER BOUND OF THE CURRENT SIZE INCREMENT IS
C                     LESS THAN THE HOLE SIZE, BRANCH TO STEP 5.
C
   10 IF(SIZE(I+1)-HOLE)20,15,15
C
C     STEP 4          ALL MATERIAL IN THE CURRENT INCREMENT IS LARGER THAN
C                     THE HOLE SIZE. NONE WILL PASS THROUGH THE HOLE. SET
C                     THE PROPORTION NOT PASSING THROUGH AT ONE AND BRANCH
C                     TO STEP 9.
C
   15 PR(I)=1.0
      GO TO 50
C
C     STEP 5          THE HOLE SIZE FALLS WITHIN THE CURRENT SIZE INCRE-
C                     MENT. COMPUTE THE PROPORTION OF MATERIAL THAT IS
C                     LARGER THAN THE HOLE SIZE AND BRANCH TO STEP 7. NOTE
C                     THAT THIS COMPUTATION ASSUMES THAT THE MATERIAL IS
C                     DISTRIBUTED RECTANGULARLY BY SIZE WITHIN THE INCRE-
C                     MENT.
C
   20 FRAC=(SIZE(I)-HOLE)/(SIZE(I)-SIZE(I+1))
      SMAX=HOLE
      GO TO 30
C
C     STEP 6          ALL THE MATERIAL IN THE CURRENT INCREMENT IS SMALLER
C                     THAN THE HOLE SIZE. SET THE PROPORTION OF THE MATER-
C                     IAL THAT IS LARGER THAN THE HOLE SIZE TO ZERO.
C
   25 FRAC=0.0
      SMAX=SIZE(I)
C
C     STEP 7          COMPUTE THE PROPORTION OF THE SMALLER MATERIAL WHICH
C                     WILL NOT PASS THROUGH THE HOLE.
C
   30 SMID=(SMAX+SIZE(I+1))/2.0
      C=1./EXP(AA*(1.0-SMID/HOLE))
C
C     STEP 8          THE OVERALL RATIO NOT PASSING THROUGH EQUALS THE
C                     PROPORTION LARGER PLUS THE PROPORTION SMALLER TIMES
C                     THE PROPORTION OF THE SMALLER MATERIAL NOT PASSING
C                     THROUGH.
C
      PR(I)=FRAC+(1.0-FRAC)*C
C
C     STEP 9          INCREMENT THE DUMMY COUNTER WHICH MOVES THROUGH THE
C                     SIZE INCREMENTS BY ONE, AND IF ALL HAVE NOT BEEN
```

FIGURE II-5-2.1. (Continued)

```
C                  PROCESSED LOOP BACK TO STEP 2.
C
   50 CONTINUE
C
C     STEP 10      END LOCAL PROCESSING.
C
      RETURN
      END
```

FIGURE II-5-2.1. (Continued)

5.3 Subroutine SCREEN

Subroutine SCREEN performs the calculations required to simulate the screening unit operation. Figure II-5-3.1 contains the listing and basic documentation for subroutine SCREEN.

Reproduced from best available copy.

```
      SUBROUTINE SCREEN(FEED,UPPER,LOWER,EQUIP,SIZE,FU,FL,NSIZE,NGRAV
     1                 ,NCHAR,D1)
C
C     TITLE(SIMULATE SCREENING UNIT OPERATION)
C
C     SUBROUTINE SCREEN IS USED TO CONTROL THE SIMULATION OF THE SCREEN-
C     ING UNIT OPERATION.
C
C     PARAMETER       DESCRIPTION
C     FEED            THIS 3-DIMENSIONAL REAL ARRAY CONTAINS THE DETAILED
C                     CHARACTERISTICS OF THE FEED TO THE SCREEN.
C     UPPER           THIS 3-DIMENSIONAL REAL ARRAY RETURNS THE DETAILED
C                     DESCRIPTION OF THE UPPER OUTPUT FLOW FROM THE SCREEN.
C     LOWER           THIS REAL 3-DIMENSIONAL ARRAY RETURNS THE DETAILED
C                     DESCRIPTION OF THE LOWER OUTPUT FLOW FROM THE SCREEN.
C     EQUIP           THIS INTEGER 1-DIMENSIONAL ARRAY CONTAINS THE EQUIP-
C                     MENT CONTROL INFORMATION FOR THE SCREEN.
C     SIZE            THIS 1-DIMENSIONAL REAL ARRAY CONTAINS THE SIZE FRAC-
C                     DISTRIBUTION.
C     FU              THIS REAL VARIABLE RETURNS THE AMOUNT OF MATERIAL
C                     IN THE UPPER OUTPUT FLOW.
C     FL              THIS REAL VARIABLE RETURNS THE AMOUNT OF MATERIAL IN
C                     THE LOWER OUTPUT FLOW.
C     NSIZE           THIS INTEGER VARIABLE CONTAINS THE NUMBER OF SIZE
C                     FRACTIONS CLASSIFYING THE FLOWSTREAMS.
C     NGRAV           THIS INTEGER VARIABLE CONTAINS THE NUMBER OF SPECIFIC
C                     GRAVITY FRACTIONS CLASSIFYING THE FLOWSTREAMS.
C     NCHAR           THIS INTEGER VARIABLE CONTAINS ONE PLUS THE NUMBER
C                     CHARACTERISTICS DEFINED FOR THE FLOWSTREAMS.
C     D1              THIS REAL VARIABLE CONTAINS THE VALUE OF THE FIRST
C                     DECISION VARIABLE FOR THE UNIT. IN THIS CASE IT IS
C                     THE SCREENING MESH SIZE.
C
C     EXTERNALS       INTS,NOTPTH,DSTRB,SDSIZE
C
C     COMMON          VARIABLES USED
C     /BLK1/          SYLD,BTURFC,SFLT,SINK,ISCC,ISRF,ISFD,SUMRY
C     /WORK/          IPTU
C     //              WS
C
C     VARIABLE        DESCRIPTION
C     ICURVE          THIS INTEGER VARIABLE CONTAINS THE BASE ADDRESS IN
C                     WORKING STORAGE OF THE CURVE CONTROL BLOCK DEFINING
C                     THE SCREENING DISTRIBUTION.
C     INCR            THIS ONE-DIMENSIONAL INTEGER ARRAY GIVES THE
C                     INCREMENTS OF THE DISTRIBUTION CURVE DIMENSIONS.
C     IX              THIS INTEGER VARIABLE CONTAINS THE ADDRESS IN WORKING
C                     STORAGE ON THE INDEPENDENT VALUES ASSOCIATED WITH THE
C                     SCREENING DISTRIBUTION.
```

FIGURE II-5-3.1. LISTING FOR SUBROUTINE SCREEN

```
C     IY              THIS INTEGER VARIABLE CONTAINS THE ADDRESS IN WORKING
C                     STORAGE OF THE DEPENDENT VALUES OF THE SCREENING
C                     DISTRIBUTION.
C     AA              THIS REAL VARIABLE CONTAINS THE EMPIRICAL SCREENING
C                     CONSTANT FROM THE DISTRIBUTION.
C     S               THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                     MOVEMENT THROUGH SIZE FRACTIONS.
C     NS              THIS INTEGER VARIABLE CONTAINS THE ROW NUMBER IN THE
C                     VARIOUS SUMMARY ARRAYS OF THE VALUES FOR THE COMPO-
C                     SITE.
C     ISRB, ISFB      THESE DUMMY INTEGER VARIABLES ARE USED FOR
C                     TEMPORARY STORAGE.
C
      DIMENSION FEED(NSIZE,NGRAV,NCHAR),UPPER(NSIZE,NGRAV,NCHAR)
     1          ,LOWER(NSIZE,NGRAV,NCHAR),EQUIP(4),SIZE(NSIZE),INCR(3)
      COMMON/WORK/NEXT,LAST,ICURVE,IUNIT,IDATA,ITITLE,ICHAR,IFLW,IASH,
     1 ISUL,IBTU
      COMMON/BLK1/ISFD,ISCC,ISRF,ISMX,SYLD(24),TYLD(24),EFFIC(24),
     1 BTUREC(24),ASHERR(24),SFLT(24),SINK(24),SMISPL(24),SNRGR(24),
     2 SGRAV(24),SPE(24),SIMP(24),SEA(24),SUMRY(24,40)
      COMMON WS(6000)
      REAL LOWER
      INTEGER WS,EQUIP,S
      DATA INCR(1),INCR(2),INCR(3)/1,0,0/
C
C     STEP 1          OBTAIN THE ADDRESSES IN WORKING STORAGE OF THE VALUES
C                     DEFINING THE SCREEN DISTRIBUTION.
C
      ICURVE=EQUIP(4)
      IX=WS(ICURVE+3)
      IY=WS(ICURVE+4)
C
C     STEP 2          USING SUBROUTINE  I N T S  OBTAIN THE VALUE OF THE
C                     SCREENING CONSTANT CORRESPONDING TO THE MESH SIZE.
C
      CALL INTS(WS(IX),WS(IY),D1,AA,WS(ICURVE+1))
C
C     STEP 3          USING SUBROUTINE  N O T P T H  COMPUTE THE CLASSIFICA-
C                     TION CURVE FOR THE SCREEN.
C
      CALL NOTPTH(SIZE,SYLD,D1,AA,NSIZE)
C
C     STEP 4          USING SUBROUTINE  D S T R B  DISTRIBUTE THE FEED
C                     AMONGST THE TWO OUTPUT STREAMS.
C
      CALL DSTRB(FEED,UPPER,LOWER,SYLD,INCR,NSIZE,NGRAV,NCHAR)
C
C     STEP 5          COMPUTE THE ROW NUMBER IN THE VARIOUS SUMMARY ARRAYS
C                     OF THE COMPOSITE SIZE FRACTION VALUES.
```

FIGURE II-5-3.1. (Continued)

```
C
      NS=NSIZE+1
C
C     STEP 6        USING SUBROUTINE  S D S I Z E  COMPUTE THE SUMMARY
C                   VALUES FOR THE FLOWSTREAMS.
C
      CALL SDSIZE(UPPER,SUMRY(1,ISCC),NSIZE,NGRAV,NCHAR,FU)
      CALL SDSIZE(LOWER,SUMRY(1,ISRF),NSIZE,NGRAV,NCHAR,FL)
C
C     STEP 7        COMPUTE THE PERCENT OF MATERIAL IN THE FEED REPORTING
C                   TO THE UNDERFLOW.
C
      DO 40 S=1,NSIZE
      SYLD(S)=(1.0-SYLD(S))*100.0
      BTUREC(S)=SYLD(S)
   40 CONTINUE
C
C     STEP 8        COMPUTE THE COMPOSITE VALUES OF PERCENT TO UNDERFLOW
C                   AND BTU TO UNDERFLOW.
C
      SYLD(NS)=100.0*FL/(FL+FU)
      ISRB=ISRF+IBTU-1
      ISFB=ISFD+IBTU-1
      BTUREC(NS)=SUMRY(NS,ISRB)*FL/SUMRY(NS,ISFB)*100.0/(FU+FL)
C
C     STEP 9        COMPUTE THE VALUES FOR OVERSIZE IN UNDEFLOW AND
C                   UNDERSIZE IN OVERFLOW.
C
      DO 45 S=1,NS
      SFLT(S)=0.0
      SINK(S)=0.0
   45 CONTINUE
      DO 55 S=1,NSIZE
      IF(SIZE(S+1).GE.D1)GO TO 50
      SFLT(S)=SUMRY(S,ISCC)
      SFLT(NS)=SFLT(NS)+SFLT(S)
      GO TO 55
   50 SINK(S)=SUMRY(S,ISRF)
      SINK(NS)=SINK(NS)+SINK(S)
   55 CONTINUE
C
C     STEP 10       END LOCAL PROCESSING.
C
      RETURN
      END
```

FIGURE II-5-3.1. (Continued)

5.4 Subroutine DCWASH

As discussed several times previously, the most difficult aspect of the washing and two-stage washing unit operations is the calculation of the distribution curves themselves. Once these curves are given, the remaining calculations are simple. The calculation of the distribution curve is complicated by the fact that the curves from which they are calculated do not necessarily have the same size distribution as the flowstream being washed. Subroutine DCWASH calculates the distribution. The key parameter passed to it is CURVE which contains the curve set control block for the curves upon which the calculation is to be based. The reader is referred to Chapter 2 of this volume for the storage conventions used. Figure II-5-4.1 contains the listing and basic documentation for subroutine DCWASH.

```
      SUBROUTINE DCWASH(DC,SIZE,GRAV,NSIZE,NGRAV,CURVE,DV)
C
C     TITLE(COMPUTE WASHER DISTRIBUTION CURVE)
C
C     SUBROUTINE DCWASH IS THE KEY ROUTINE USED BY THE OPERATIONS OF
C     WASHING AND TWO-STAGE WASHING. IT IS USED TO COMPUTE THE DISTRI-
C     BUTION CUVE FOR THE UNIT GIVEN THE VALUE OF SPECIFIC GRAVITY OF
C     SEPARATION.
C
C     EXTERNALS      VINTRP, INTS, BRSIZE
C
C     COMMON         VARIABLES USED
C     //             WS
C
C     PARAMETER      DESCRIPTION
C     DC             THIS 2-DIMENSIONAL, REAL ARRAY RETURNS THE DISTRI-
C                    BUTION CURVE VALUES FOR THE UNIT.
C     SIZE           THIS REAL 1-DIMENSIONAL ARRAY CONTAINS THE SIZE
C                    FRACTION DEFINITION FOR THE FLOWSTREAMS.
C     GRAV           THIS REAL, 1-DIMENSIONAL ARRAY CONTAINS THE SPECIFIC
C                    GRAVITY FRACTION DEFINITION FOR THE FLOWSTREAMS.
C     NSIZE          THIS INTEGER VARIABLE SPECIFIES THE NUMBER OF SIZE
C                    FRACTIONS CLASSIFYING THE FLOWSTREAMS.
C     NGRAV          THIS INTEGER VARIABLE DEFINES THE NUMBER OF SPECIFIC
C                    GRAVITY FRACTIONS DEFINING THE FLOWSTREAMS.
C     CURVE          THIS 1-DIMENSIONAL INTEGER ARRAY CONTAINS THE CURVE
C                    CONTROL INFORMATION FOR THE CURVES FROM WHICH THE
C                    DISTRIBUTION CURVE IS TO BE COMPUTED.
C     DV             THIS REAL VARIABLE CONTAINS THE VALUE OF THE DECISION
C                    VARIABLE FOR THE UNIT. THIS IS SPECIFIC GRAVITY OF
C                    SEPARATION IN THIS CASE.
C
C     VARIABLE       DESCRIPTION
C     BRG, IBRG      THIS TYPELESS, 1-DIMENSIONAL ARRAY IS USED TO CONTAIN
C                    THE BRIDGE SPECIFICATION ARRAY FOR MOVING FROM THE
C                    CURVE SIZE DISTRIBUTION TO THE FLOWSTREAN DISTRI-
C                    BUTION.
C     S              THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                    MOVEMENT THROUGH SIZE FRACTIONS.
C     G              THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                    MOVEMENT THROUGH THE SPECIFIC GRAVITY FRACTIONS.
C     NPOINT         THIS INTEGER VARIABLE IS USED TO CONTAIN THE NUMBER
C                    OF POINTS IN THE CURVE.
C     ISC            THIS INTEGER VARIABLE CONTAINS THE ADDRESS IN WORK-
C                    ING STORAGE OF THE CURVE SIZE FRACTION DEFINITION.
C     NSC            THIS INTEGER VARIABLE CONTAINS THE NUMBER OF SIZE
C                    FRACTIONS OVER WHICH THE CURVE IS DEFINED.
C     NBRG           THIS INTEGER VARIABLE CONTAINS THE NUMBER OF CELLS
C                    USED IN THE BRIDGE SPECIFICATION ARRAY.
```

FIGURE II-5-4.1. LISTING FOR SUBROUTINE DCWASH

```
C                   ENT VARIABLE FOR LOOK-UP IN THE CURVE.
C      B            THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                   MOVEMENT THROUGH THE BRIDGE SPECIFICATION ARRAY.
C      IX           THIS INTEGER VARIABLE CONTAINS THE ADDRESS IN WORKING
C                   STORAGE OF THE INDEPENDENT CURVE POINTS.
C      IY           THIS INTEGER VARIABLE CONTAINS THE ADDRESS IN WORKING
C                   STORAGE OF THE INDEPENDENT CURVE POINTS.
C      SC           THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                   MOVEMENT THROUGH THE CURVE SIZE FRACTIONS.
C      NS           THIS INTEGER VARIABLE CONTAINS THE NUMBER OF FLOW-
C                   STREAM SIZE FRACTIONS TO WHICH A GIVEN CURVE SIZE
C                   FRACTION CONTRIBUTES.
C      YVAL         THIS REAL VARIABLE CONTAINS THE INDEPENDENT VALUE
C                   COMPUTED FROM THE CURVE VIA LOOK-UP.
C      SB           THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                   MOVEMENT THROUGH THE SIZE FRACTIONS TO WHICH A
C                   GIVEN CURVE FRACTION CONTRIBUTES.
C      INW          THIS INTEGER VARIABLE CONTAINS THE ADDRESS IN WORKING
C                   STORAGE OF THE COUNT CELLS ASSOCITED WITH EACH SIZE
C                   FRACTION FOR A SET OF EXPLICIT PERFORMANCE CURVES.
C      LW           THIS INTEGER VARIABLE CONTAINS THE ENDING ADDRESS
C                   OF THE VALUES ASSOCIATED WITH EACH OF THE ACTUAL PER-
C                   FORMANCE CURVES
C      XW, NW       THIS TYPELESS VARIABLE CONTAINS THE NUMBER OF CURVES
C                   DEFINED FOR THE CURRENT CURVE SIZE FRACTION.
C      LB           THIS INTEGER VARIABLE CONTAINS THE START IN THE
C                   BRIDGE SPECIFICATION ARRAY OF THE INFORMATION FOR
C                   THE CURRENT CURVE SIZE FRACTION.
C      IWASH        THIS INTEGER VARIABLE CONTAINS THE ADDRESS IN
C                   WORKING STORAGE OF THE VALUE ASSOCIATED WITH THE
C                   FIRST CURVE FROM WHICH THE ACTUAL CURVE IS TO BE
C                   COMPUTED.
C      LWASH        THIS INTEGER VARIABLE CONTAINS THE ADDRESS IN WORKING
C                   STORAGE OF THE VALUE ASSOCIATED WITH THE LAST CURVE
C                   UPON WHICH THE ACTUAL CURVE IS TO BE BASED.
C      IXA          THIS INTEGER VARIABLE CONTAINS THE ADDRESS IN
C                   WORKING STORAGE OF THE INDEPENDENT POINTS ASSOCIATED
C                   WITH THE FIRST CURVE UPON WHICH THE ACTUAL CURVE IS
C                   TO BE BASED.
C      LXA          THIS INTEGER VARIABLE CONTAINS THE ADDRESS IN
C                   WORKING STORAGE OF THE INDEPENDENT POINTS ASSOCIATED
C                   WITH THE LAST CURVE UPON WHICH THE ACTUAL CURVE IS
C                   TO BE BASED.
C      IYA          THIS INTEGER VARIABLE CONTAINS THE ADDRESS IN
C                   WORKING STORAGE OF THE DEPENDENT POINTS ASSOCIATED
C                   WITH THE FIRST CURVE UPON WHICH THE ACTUAL CURVE IS
C                   TO BE BASED.
C      LYA          THIS INTEGER VARIABLE CONTAINS THE ADDRESS IN
C                   WORKING STORAGE OF THE DEPENDENT POINTS ASSOCIATED
```

FIGURE II-5-4.1. (Continued)

```
C                      WITH THE LAST CURVE UPON WHICH THE ACTUAL CURVE IS
C                      TO BE BASED.
C      RATIO           THIS REAL VARIABLE CONTAINS THE RATIO USED FOR ALLOC-
C                      TING THE VALUES COMPUTED FROM TWO ACTUAL CURVES.
C      YMIN            THIS REAL VARIABLE CONTAINS THE LOOK-UP VALUE FROM THI
C                      LAST ACTUAL CURVE.
C      XVAL            THIS REAL VARIABLE CONTAINS THE ARGUMENT TO THE
C                      DISTRIBUTION CURVES.
C      IW              THIS INTEGER VARIABLE CONTAINS THE STARTING ADDRESS
C                      OF THE VALUES ASSOCIATED WITH EACH OF THE ACTUAL
C                      PERFORMANCE CURVES.
C
      DIMENSION DC(24,20),SIZE(1),GRAV(NGRAV),CURVE(6)
      COMMON WS(6000)
      INTEGER CURVE,S,G,SC,SB,B
      COMMON/LOCALV/BRG(100),NPOINT,ISC,NBRG,B,IX,IY,NSC,NS,YVAL,INW
     1               ,LW,XW,LB,IWASH,LWASH,IXA,LXA,IYA,LYA,RATIO,YMIN
     2               ,XVAL,IW,LVFILL(72)
      DIMENSION IBRG(100)
      EQUIVALENCE (BRG(1),IBRG(1)),(XW,NW)
C
C     STEP 1          ZERO OUT THE CONTENTS OF THE ARRAY TO RETURN THE
C                     FINAL COMPUTED DISTRIBUTION CURVE VALUES.
C
      DO 10 S=1,24
      DO 10 G=1,20
      DC(S,G)=0.0
   10 CONTINUE
C
C     STEP 2          OBTAIN THE NUMBER OF SIZE FRACTIONS OVER WHICH
C                     THE CURVES ARE DEFINED AND THE ADDRESS IN WORKING
C                     STORAGE OF THE SIZE FRACTION DEFINITION.
C
      NPOINT=CURVE(2)
      ISC=CURVE(6)
      NSC=CURVE(3)
C
C     STEP 3          USING SUBROUTINE  B R S I Z E  COMPUTE THE SPECIFIC-
C                     ATION NEEDED TO MOVE FROM THE CURVE SIZE DISTRIBUTION
C                     TO THE FLOWSTREAM SIZE DISTRIBUTION.
C
      CALL BRSIZE(SIZE,NSIZE,WS(ISC),NSC,BRG,NBRG)
C
C     STEP 4          IF THE DISTRIBUTION CURVE IS TO BE CALCULATED FROM
C                     AN EXPLICITE SET OF PERFORMANCE CURVES, BRANCH TO
C                     STEP 11.
C
      IF(CURVE(1).EQ.5)GO TO 30
C
```

FIGURE II-5-4.1. (Continued)

```
C     STEP 5         INITIALIZE AT ONE THE DUMMY COUNTER USED TO
C                    CONTROL THE MOVEMENT THROUGH THE SPECIFIC GRAVITY
C                    FRACTIONS.
C
      DO 25 G=1,NGRAV
C
C     STEP 6         COMPUTE THE VALUE OF C-REDUCED SPECIFIC GRAVITY FOR
C                    LOOK-UP IN THE C-GENERALIZED DISTRIBUTION CURVES.
C
      XVAL=GRAV(G)/DV
C
C     STEP 7         INITIALIZE THE VARIABLES NEEDED TO MOVE THROUGH
C                    THE CURVE SIZE DISTRIBUTION.
C
      B=0
      IX=CURVE(4)-NPOINT
      IY=CURVE(5)-NPOINT
C
C     STEP 8         MOVING THROUGH THE SIZE FRACTIONS COMPUTE THE
C                    DISTRIBUTION CURVE VALUES FOR THE CURRENT SPECIFIC
C                    GRAVITY FRACTION. SUBROUTINE  Y I N T R P  IS USED
C                    TO PERFORM THE ACTUAL LOOK-UP.
C
      DO 20 SC=1,NSC
      B=B+1
      NS=IBRG(B)
      IX=IX+NPOINT
      IY=IY+NPOINT
      IF(NS.EQ.0)GO TO 20
      YVAL=YINTRP(WS(IX),WS(IY),XVAL,NPOINT)
      DO 15 SB=1,NS
      B=B+1
      S=IBRG(B)
      B=B+1
      DC(S,G)=DC(S,G)+YVAL*BRG(B)
   15 CONTINUE
   20 CONTINUE
C
C     STEP 9         INCREMENT BY ONE THE DUMMY COUNTER CONTROLING THE
C                    MOVEMENT THROUGH THE SPECIFIC GRAVITY FRACTIONS, AND
C                    IF ALL HAVE NOT BEEN PROCESSED, LOOP BACK TO STEP 6.
C
   25 CONTINUE
C
C     STEP 10        END LOCAL PROCESSING.
C
      GO TO 999
C
C     STEP 11        THE DISTRIBUTION CURVE IS TO BE COMPUTED FROM
```

FIGURE II-5-4.1. (Continued)

```
C                     EXPLICIT PERFORMANCE CURVES. BEGIN BY INITIALIZING
C                     THE VARIABLES NEEDED TO MOVE THROUGH THE CURVES.
C
   30 B=0
      IX=CURVE(4)
      IY=CURVE(5)
      INW=ISC+NSC
      LW=INW+NSC
C
C     STEP 12         INITIALIZE AT ONE THE DUMMY COUNTER USED TO CONTROL
C                     THE MOVEMENT THROUGH THE CURVE SIZE FRACTIONS.
C
      DO 80 SC=1,NSC
C
C     STEP 13         INITIALIZE THE VARIABLES NEEDED TO MOVE THROUGH THE
C                     CURVES SUPPLIED FOR THE CURRENT SIZE FRACTION.
C
      IW=LW+1
      INW=INW+1
      XW=WS(INW)
      LW=LW+NW
C
C     STEP 14         OBTAIN THE NUMBER OF SIZE FRACTIONS TO WHICH THE
C                     NEXT CURVE SIZE CONTRIBUTES, AND IF IT IS ZERO,
C                     BRANCH TO STEP 17.
C
      B=B+1
      NS=IBRG(B)
      IF(NS.EQ.0)GO TO 75
C
C     STEP 15         FIND THE ACTUAL PERFORMANCE CURVE OR PAIR OF PER-
C                     FORMANCE CURVES, WHICH ACTUALLY RELATE TO THE UNIT.
C                     THIS DETERMINATION IS MADE FROM THE VALUE OF THE
C                     DECISION VARIABLE FOR THE UNIT.
C
      LB=B
      IWASH=IW
      IF(IW.GE.LW)GO TO 50
   35 IF(DV-WS(IWASH))45,50,40
   40 IF(IWASH.EQ.LW)GO TO 50
      IWASH=IWASH+1
      GO TO 35
   45 IF(IWASH.EQ.IW)GO TO 50
      LWASH=IWASH-1
      RATIO=(DV-WS(LWASH))/(WS(IWASH)-WS(LWASH))
      GO TO 55
   50 LWASH=IWASH
C
C     STEP 16         COMPUTE THE ACTUAL DISTRIBUTION VALUES FROM THE
```

FIGURE II-5-4.1. (Continued)

```
C                   CURVE SIZE FRACTION. SUBROUTINE  I N T S  IS USED TO
C                   PERFORM THE ACTUAL LOOK-UPS.
C
 55   IXA=IX+(IWASH-IW)*NPOINT
      IYA=IY+IXA-IX
      LXA=IXA-NPOINT
      LYA=IYA-NPOINT
      DO 70 G=1,NGRAV
      CALL INTS(WS(IXA),WS(IYA),GRAV(G),YVAL,NPOINT)
      IF(IWASH.EQ.LWASH)GO TO 60
      CALL INTS(WS(LXA),WS(LYA),GRAV(G),YMIN,NPOINT)
      YVAL=YMIN+RATIO*(YVAL-YMIN)
 60   B=LB
      DO 65 SB=1,NS
      B=B+1
      S=IBRG(B)
      B=B+1
      DC(S,G)=DC(S,G)+YVAL*BRG(B)
 65   CONTINUE
 70   CONTINUE
C
C     STEP 17       ALL PROCESSING HAS BEEN PERFORMED ON THE CURRENT
C                   CURVE SIZE FRACTION. UPDATE THE VARIABLES.
C
 75   IX=IX+NW*NPOINT
      IY=IY+NW*NPOINT
C
C     STEP 18       INCREMENT BY ONE THE DUMMY COUNTER CONTROLLING
C                   THE MOVEMENT THROUGH THE CURVE SIZE FRACTIONS, AND
C                   IF ALL HAVE NOT BEEN PROCESSED, LOOP BACK TO STEP 13.
C
 80   CONTINUE
C
C     STEP 19       END LOCAL PROCESSING.
C
999   RETURN
      END
```

FIGURE II-5-4.1. (Continued)

5.5 Subroutine WASH

The unit operation of washing is implemented in subroutine WASH. Given subroutine DCWASH, this routine is straightforward. Figure II-5-5.1 contains the listing and basic documentation for subroutine WASH.

```
      SUBROUTINE WASH(FEED,CLEAN,REFUSE,EQUIP,SIZE,GRAV,DISTRB,NSIZE
     1               ,NGRAV,NCHAR,SGSP)
C
C     TITLE(SIMULATE THE WASHING UNIT OPERATION)
C
C     SUBROUTINE WASH IS THE MAIN CONTROL ROUTINE USED TO SIMULATE THE
C     WASHING UNIT OPERATION.
C
C     PARAMETER    DESCRIPTION
C     FEED          THIS 3-DIMENSIONAL, REAL ARRAY CONTAINS THE VALUES
C                   FOR THE FEED STREAM INTO THE WASHING DEVICE.
C     CLEAN         THIS 3-DIMENSIONAL, REAL ARRAY RETURNS THE VALUES
C                   FOR THE CLEAN OUTPUT FLOW PRODUCED BY THE WASHING
C                   UNIT.
C     REFUSE        THIS 3-DIMENSIONAL, REAL ARRAY RETURNS THE VALUES
C                   FOR THE REFUSE STREAM PRODUCED BY THE WASHING UNIT.
C     EQUIP         THIS 1-DIMENSIONAL INTEGER ARRAY CONTAINS THE
C                   EQUIPMENT TYPE CONTROL INFORMATION FOR THE UNIT.
C     SIZE          THIS 1-DIMENSIONAL, REAL ARRAY CONTAINS THE SIZE
C                   FRACTION DEFINITION FOR THE FLOWSTREAMS.
C     GRAV          THIS 1-DIMENSIONAL, REAL ARRAY CONTAINS THE SPECIFIC
C                   GRAVITY FRACTION DEFINITION FOR THE FLOWSTREAMS.
C     DISTRB        THIS 2-DIMENSIONAL, REAL ARRAY IS USED TO STORE
C                   THE DISTRIBUTION CURVE COMPUTED FOR THE UNIT.
C     NSIZE         THIS INTEGER VARIABLE CONTAINS THE NUMBER OF SIZE
C                   FRACTIONS DEFINING THE FLOWSTREAMS.
C     NGRAV         THIS INTEGER VARIABLE CONTAINS THE NUMBER OF
C                   SPECIFIC GRAVITY FRACTIONS DEFINING THE FLOWSTREAMS.
C     NCHAR         THIS INTEGER VARIABLE CONTAINS ONE PLUS THE NUMBER
C                   OF CHARACTERISTICS DEFINED FOR THE FLOWSTREAMS.
C     SGSP          THIS REAL VARIABLE CONTAINS THE VALUE FOR SPECIFIC
C                   GRAVITY OF SEPARATION TO BE USED TO COMPUTE THE
C                   DISTRIBUTION CURVE FOR THE UNIT.
C
C     COMMON       VARIABLES USED
C     //           WS
C
C     EXTERNALS    DCWASH, DSTRB
C
C     VARIABLE     DESCRIPTION
C     ICURVE        THIS INTEGER VARIABLE CONTAINS THE ADDRESS IN
C                   WORKING STORAGE OF THE CURVE SET DEFINING THE
C                   DISTRIBUTION CURVE FOR THE UNIT.
C     INCR          THIS 1-DIMENSIONAL INTEGER ARRAY IS USED TO SPECIFY
C                   THE INCREMENTS OF THE DISTRIBUTION CURVE DIMENSIONS.
C
      DIMENSION FEED(NSIZE,NGRAV,NCHAR),CLEAN(NSIZE,NGRAV,NCHAR)
     1          ,REFUSE(NSIZE,NGRAV,NCHAR),EQUIP(4),SIZE(NSIZE)
     2          ,GRAV(NGRAV),DISTRB(24,20)
```

FIGURE II-5-5.1. LISTING FOR SUBROUTINE WASH

```
      DIMENSION INCR(3)
      COMMON WS(6000)
      INTEGER EQUIP,WS
      DATA INCR(1),INCR(2),INCR(3)/24,20,0/
C
C     STEP 1          OBTAIN THE ADDRESS IN WORKING STORAGE OF THE CURVE
C                     SET DEFINING THE DISTRIBUTION CURVE FROM THE FOURTH
C                     CELL OF THE EQUIPMENT TYPE CONTROL VECTOR.
C
      ICURVE=EQUIP(4)
C
C     STEP 2          USING SUBROUTINE  D C W A S H  COMPUTE THE ACTUAL
C                     DISTRIBUTION CURVE FOR THE UNIT.
C
      CALL DCWASH(DISTRB,SIZE,GRAV,NSIZE,NGRAV,WS(ICURVE),SGSP)
C
C     STEP 3          USING SUBROUTINE  D S T R B  COMPUTE THE VALUES FOR
C                     THE CLEAN AND REFUSE OUTPUT FLOWS BASED ON THE
C                     DISTRIBUTION CURVE.
C
      CALL DSTRB(FEED,CLEAN,REFUSE,DISTRB,INCR,NSIZE,NGRAV,NCHAR)
C
C     STEP 4          END LOCAL PROCESSING.
C
      RETURN
      END
```

FIGURE II-5-5.1. (Continued)

5.6 Subroutine WASH2

Two-stage washing is implemented via subroutine WASH2. As with WASH, given the operation of DCWASH the calculations in WASH2 are simple. Figure II-5-6.1 gives the listing and basic documentation for subroutine WASH2.

```
      SUBROUTINE WASH2(FEED,CLEAN,MIDDLE,REFUSE,EQUIP,SIZE,GRAV,DISTRB
     1                ,NSIZE,NGRAV,NCHAR,SGSP)
C
C     TITLE(SIMULATE THE 2-STAGE WASHING UNIT OPERATION)
C
C     SUBROUTINE WASH2 IS THE MAIN CONTROL ROUTINE USED TO SIMULATE THE
C     2-STAGE WASHING OPERATION.
C
C     PARAMETER     DESCRIPTION
C     FEED          THIS 3-DIMENSIONAL, REAL ARRAY CONTAINS THE VALUES
C                   FOR THE FEED STREAM INTO THE WASHING DEVICE.
C     CLEAN         THIS 3-DIMENSIONAL, REAL ARRAY RETURNS THE VALUES
C                   FOR THE CLEAN OUTPUT FLOW PRODUCED BY THE WASHING
C                   UNIT.
C     MIDDLE        THIS 3-DIMENSIONAL, REAL ARRAY RETURNS THE VALUES
C                   FOR THE MIDDLINGS STREAM PRODUCED BY THE UNIT.
C     REFUSE        THIS 3-DIMENSIONAL, REAL ARRAY RETURNS THE VALUES
C                   FOR THE REFUSE STREAM PRODUCED BY THE WASHING UNIT.
C     EQUIP         THIS 1-DIMENSIONAL INTEGER ARRAY CONTAINS THE
C                   EQUIPMENT TYPE CONTROL INFORMATION FOR THE UNIT.
C     SIZE          THIS 1-DIMENSIONAL, REAL ARRAY CONTAINS THE SIZE
C                   FRACTION DEFINITION FOR THE FLOWSTREAMS.
C     GRAV          THIS 1-DIMENSIONAL, REAL ARRAY CONTAINS THE SPECIFIC
C                   GRAVITY FRACTION DEFINITION FOR THE FLOWSTREAMS.
C     DISTRB        THIS 2-DIMENSIONAL, REAL ARRAY IS USED TO STORE
C                   THE DISTRIBUTION CURVE COMPUTED FOR THE UNIT.
C     NSIZE         THIS INTEGER VARIABLE CONTAINS THE NUMBER OF SIZE
C                   FRACTIONS DEFINING THE FLOWSTREAMS.
C     NGRAV         THIS INTEGER VARIABLE CONTAINS THE NUMBER OF
C                   SPECIFIC GRAVITY FRACTIONS DEFINING THE FLOWSTREAMS.
C     NCHAR         THIS INTEGER VARIABLE CONTAINS ONE PLUS THE NUMBER
C                   OF CHARACTERISTICS DEFINED FOR THE FLOWSTREAMS.
C     SGSP          THIS REAL VARIABLE CONTAINS THE VALUE FOR SPECIFIC
C                   GRAVITY OF SEPARATION TO BE USED TO COMPUTE THE
C                   DISTRIBUTION CURVE FOR THE UNIT.
C
C     COMMON        VARIABLES USED
C     //            WS
C
C     EXTERNALS     DCWASH, DSTRB
C
C     VARIABLE      DESCRIPTION
C     ICURVE        THIS INTEGER VARIABLE CONTAINS THE ADDRESS IN
C                   WORKING STORAGE OF THE CURVE SET DEFINING THE
C                   DISTRIBUTION CURVE FOR THE UNIT.
C     S             THIS DUMMY, INTEGER VARIABLE IS USED TO CONTROL THE
C                   MOVEMENT THROUGH THE SIZE FRACTIONS.
C     G             THIS DUMMY, INTEGER VARIABLE IS USED TO CONTROL THE
C                   MOVEMENT THROUGH THE SPECIFIC GRAVITY FRACTIONS.
```

FIGURE II-5-6.1. LISTING FOR SUBROUTINE WASH2

```
C     C            THIS DUMMY, INTEGER VARIABLE IS USED TO CONTROL THE
C                  MOVEMENT THROUGH THE FLOW CHARACTERISTICS.
C     INCR         THIS 1-DIMENSIONAL INTEGER ARRAY IS USED TO SPECIFY
C                  THE INCREMENTS IN THE DISTRIBUTION CURVE DIMENSIONS.
C
      DIMENSION FEED(NSIZE,NGRAV,NCHAR),CLEAN(NSIZE,NGRAV,NCHAR)
     1          ,MIDDLE(NSIZE,NGRAV,NCHAR),REFUSE(NSIZE,NGRAV,NCHAR)
     1          ,EQUIP(4),SIZE(NSIZE),GRAV(NGRAV),DISTRB(24,20)
      DIMENSION INCR(3)
      COMMON WS(6000)
      INTEGER S,G,C,EQUIP,WS
      REAL MIDDLE
      DATA INCR(1),INCR(2),INCR(3)/24,20,0/
C
C     STEP 1       OBTAIN THE ADDRESS IN WORKING STORAGE OF THE CURVE
C                  SET DEFINING THE MIDDLINGS DISTRIBUTION CURVE FROM
C                  THE FOURTH CELL OF THE EQUIPMENT TYPE CONTROL VECTOR.
C
      ICURVE=EQUIP(4)+6
C
C     STEP 2       USING SUBROUTINE  D C W A S H  COMPUTE THE DISTRI-
C                  BUTION CURVE FOR THE MIDDLINGS.
C
      CALL DCWASH(DISTRB,SIZE,GRAV,NSIZE,NGRAV,WS(ICURVE),SGSP)
C
C     STEP 3       USING SUBROUTINE  D S T R B  COMPUTE THE VALUES FOR
C                  THE MIDDLINGS AND REFUSE OUTPUT FLOWS
C
      CALL DSTRB(FEED,MIDDLE,REFUSE,DISTRB,INCR,NSIZE,NGRAV,NCHAR)
C
C     STEP 4       COMPUTE THE ADDRESS IN WORKING STORAGE OF THE CURVE
C                  SET DEFINING THE DISTRIBUTION CURVE FOR THE CLEAN
C                  OUTPUT FLOW.
C
      ICURVE=ICURVE-6
C
C     STEP 5       USING SUBROUTINE  D C W A S H  CALCULATE THE DISTRI-
C                  BUTION CURVE FOR THE CLEAN OUTPUT FLOW.
C
      CALL DCWASH(DISTRB,SIZE,GRAV,NSIZE,NGRAV,WS(ICURVE),SGSP)
C
C     STEP 6       MOVING THROUGH THE SIZE AND SPECIFIC GRAVITY FRAC-
C                  TIONS COMPUTE THE VALUES FOR THE CLEAN AND REFUSE
C                  OUTPUT FLOWS.
C
      DO 30 S=1,NSIZE
      DO 30 G=1,NGRAV
      DO 30 C=1,NCHAR
      CLEAN(S,G,C)=FEED(S,G,C)*DISTRB(S,G)
```

FIGURE II-5-6.1. (Continued)

```
      REFUSE(S,G,C)=REFUSE(S,G,C)-CLEAN(S,G,C)
      IF(REFUSE(S,G,C).GE.0.0)GO TO 30
      MIDDLE(S,G,C)=MIDDLE(S,G,C)+REFUSE(S,G,C)
      REFUSE(S,G,C)=0.0
   30 CONTINUE
      RETURN
      END
```

FIGURE II-5-6.1. (Continued)

5.7 Subroutine FROTH

The froth flotation unit operation is implemented via subroutine FROTH. Figure II-5-7.1 contains the listing for that routine.

```
      SUBROUTINE FROTH(FEED,CC,REF,SIZE,GRAV,GBOUND,NSIZE,NGRAV,NCHAR
     1                 ,EQUIP)
C
C     TITLE(SIMULATE FROTH FLOTATION UNIT OPERATION)
C
C     SUBROUTINE FROTH IS USED TO CONTROL THE SIMULATION OF THE FROTH
C     FLOTATION UNIT OPERATION.
C
C     PARAMETER      DESCRIPTION
C     FEED           THIS 3-DIMENSIONAL REAL ARRAY CONTAINS THE DISTRIBU-
C                    TION AND CHARACTERISTICS OF THE FEED TO THE UNIT.
C     CC             THIS 3-DIMENSIONAL REAL ARRAY RETURNS THE DISTRIBU-
C                    TION AND CHARACTERISTICS OF THE CLEAN OUTPUT FLOW
C                    FROM THE UNIT.
C     REF            THIS 3-DIMENSIONAL REAL ARRAY RETURNS THE DISTRIBU-
C                    TION AND CHARACTERISTICS OF THE REFUSE OUTPUT FLOW
C                    FROM THE UNIT.
C     SIZE           THIS 1-DIMENSIONAL REAL ARRAY DEFINES THE FLOWSTREAM
C                    SIZE DISTRIBUTION.
C     GRAV           THIS REAL 1-DIMENSIONAL ARRAY CONTAINS THE MIDPOINTS
C                    OF THE SPECIFIC GRAVITY FRACTIONS.
C     GBOUND         THIS REAL 1-DIMENSIONAL ARRAY CONTAINS THE SPECIFIC
C                    GRAVITY DISTRIBUTION FOR THE FLOWSTREAMS.
C     NSIZE          THIS INTEGER VARIABLE DEFINES THE NUMBER OF SIZE
C                    FRACTIONS.
C     NGRAV          THIS INTEGER VARIABLE DEFINES THE NUMBER OF SPECIFIC
C                    GRAVITY FRACTIONS.
C     NCHAR          THIS INTEGER VARIABLE DEFINES THE NUMBER OF CHARACT-
C                    ERISTICS PLUS ONE SPECIFIED FOR THE FLOWSTREAMS.
C     EQUIP          THIS 1-DIMENSIONAL INTEGER ARRAY CONTAINS THE EQUIP-
C                    MENT INFORMATION BLOCK FOR THE UNIT.
C
C     COMMON         VARIABLES USED
C     /BLK1/         ISFD, SUMRY
C     /WORK/         IASH
C     //             WS
C
C     EXTERNALS      YINTRP, FLOAT, ABS
C
C     VARIABLE       DESCRIPTION
C     NS             THIS INTEGER VARIABLE CONTAINS ONE PLUS THE NUMBER
C                    OF SIZE FRACTIONS.
C     NG             THIS INTEGER VARIABLE CONTAINS ONE PLUS THE NUMBER
C                    OF SPECIFIC GRAVITY FRACTIONS.
C     I              THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                    MOVEMENT THROUGH THE SIZE FRACTIONS.
C     WT             THIS REAL VARIABLE CONTAINS THE WEIGHT OF FEED
C     SCMWT          THIS REAL, 1-DIMENSIONAL ARRAY CONTAINS THE CUMULA-
C                    TIVE WEIGHT BY SIZE FRACTION IN THE FEED.
```

FIGURE II-5-7.1. LISTING FOR SUBROUTINE FROTH

```
C       SCMASH         THIS REAL 1-DIMENSIONAL ARRAY CONTAINS THE CUMULA-
C                      TIVE WEIGHT OF ASH BY SPECIFIC GRAVITY IN THE FEED.
C       J1             THIS DUMMY INTEGER VARIABLE IS USED AS A UPPER LIMIT
C                      IN THE CALCULATION OF CUMULATIVE VALUES.
C       K              THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                      MOVEMENT THROUGH LOCAL LOOPS.
C       J              THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                      MOVEMENT THROUGH SPECIFIC GRAVITY FRACTIONS.
C       YIELD          THIS REAL, 1-DIMENSIONAL ARRAY CONTAINS THE DESIRED
C                      YIELD FOR EACH SIZE FRACTION.
C       ASH            THIS REAL, 1-DIMENSIONAL ARRAY CONTAINS THE DESIRED
C                      PROPORTION ASH FOR EACH SIZE FRACTION.
C       IJX            THIS INTEGER VARIABLE CONTAINS THE ADDRESS IN WORK-
C                      ING STORAGE OF THE X-POINT VALUES ASSOCIATED WITH
C                      THAT DISTRIBUTION CURVE WHICH COMES CLOSEST TO
C                      GIVING THE DESIRED RESULTS.
C       DMIN           THIS REAL VARIABLE CONTAINS THE MIMIMUM DIFFERENCE
C                      ENCOUNTERED BETWEEN THE SPECIFIC GRAVITY OF SEPARA-
C                      TION GIVING THE CLOSEST ASH VALUE AND THE ONE GIVING
C                      THE DESIRED YIELD VALUE.
C       ICURVE         THIS INTEGER VARIABLE CONTAINS THE ADDRESS IN WORK-
C                      ING STORAGE OF THE CURVE SET CONTROL BLOCK CONTAIN-
C                      ING THE DISTRIBUTION CURVES.
C       NCURVE         THIS INTEGER VARIABLE CONTAINS THE NUMBER OF CURVES.
C       NPOINT         THIS INTEGER VARIABLE CONTAINS THE NUMBER OF POINTS
C                      PER CURVE.
C       IX             THIS INTEGER VARIABLE CONTAINS THE ADDRESS IN WORK-
C                      ING STORAGE OF THE X-POINTS ASSOCIATED WITH THE CUR-
C                      RENT CURVE.
C       IY             THIS INTEGER VARIABLE CONTAINS THE ADDRESS IN WORK-
C                      ING STORAGE OF THE Y-POINTS ASSOCIATED WITH THE
C                      CURRENT CURVE.
C       I1             THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                      MOVEMENT THROUGH THE CURVES.
C       JK             THIS INTEGER VARIABLE CONTAINS THE SEQUENCE NUMBER
C                      OF THE CURRENT CURVE.
C       INDEX          THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                      MOVEMENT THROUGH THE 20 VALUES OF DESIRED SPECIFIC
C                      GRAVITY OF SEPARATION TO BE APPLIED TO EACH CURVE.
C       SUM1           THIS REAL VARIABLE CONTAINS THE TOTAL AMOUNT OF
C                      MATERIAL IN THE FEED IN THE CURRENT SIZE FRACTION.
C       RATIO          THIS DUMMY REAL VARIABLE IS USED IN THE COMPUTATION
C                      OF YIELD AND PERCENT ASH FOR EACH ATTEMPTED DISTRI-
C                      BUTION.
C       SUM2           THIS REAL VARIABLE CONTAINS THE AMOUNT OF MATERIAL
C                      IN CLEAN IF THE CURRENT CURVE WERE USED.
C       SUM3           THIS REAL VARIABLE CONTAINS THE AMOUNT OF ASH IN
C                      THE CLEAN IF THE CURRENT CURVE WERE USED.
C       YLDPTS         THIS REAL, 1-DIMENSIONAL ARRAY CONTAINS THE YIELD
```

FIGURE II-5-7.1. (Continued)

Reproduced from
best available copy.

```
C                 ASSOCIATED WITH THE CURRENT CURVE AND THE CURRENT
C                 DESIRED SPECIFIC GRAVITY OF SEPARATION.
C     ASHPTS      THIS REAL 1-DIMENSIONAL ARRAY CONTAINS THE PROPOR-
C                 TION ASH IN THE CLEAN IF THE CURRENT CURVE AND SEPAR-
C                 ATION GRAVITY WERE USED.
C     S           THIS REAL, 1-DIMENSIONAL ARRAY CONTAINS THE VALUES
C                 OF THE SEPARATION GRAVITIES ATTEMPTED.
C     SGSP1       THIS REAL VARIABLE CONTAINS THE SEPARATION GRAVITY
C                 GIVING THE DESIRED YIELD.
C     SGSP2       THIS REAL VARIABLE CONTAINS THE SEPARATION GRAVITY
C                 GIVING THE DESIRED PROPORTION ASH.
C     SGAV        THIS REAL VARIABLE CONTAINS THE AVERAGE OF THE
C                 ASH SEPARATION GRAVITY AND THE YIELD GRAVITY.
C     DIFF        THIS REAL VARIABLE CONTAINS THE ABSOLUTE DIFFERENCE
C                 BETWEEN THE ASH GRAVITY AND THE YIELD GRAVITY.
C     IJY         THIS INTEGER VARIABLE CONTAINS THE ADDRESS IN WORK-
C                 ING STORAGE OF THE Y-POINTS ASSOCIATED WITH THAT
C                 CURVE GIVING THE BEST RESULTS.
C     SGSP        THIS REAL VARIABLE CONTAINS THE SEPARATION GRAVITY
C                 GIVING THE BEST OVERALL RESULTS.
C     X           THIS REAL VARIABLE CONTAINS THE GENERALIZED VALUE
C                 OF SPECIFIC GRAVITY FOR LOOK-UP IN THE DISTRIBUTIONS.
C     XYLD        THIS REAL VARIABLE CONTAINS THE YIELD OBTAINED FROM
C                 THE BEST OVERALL CURVE.
C     XASH        THIS REAL VARIABLE CONTAINS THE PROPORTION ASH
C                 OBTAINED FROM THE BEST OVERALL CURVE.
C     YLDAV       THIS REAL VARIABLE CONTAINS THE AVERAGE OF THE BEST
C                 OVERALL YIELD AND THE DESIRED YIELD.
C     ASHAV       THIS REAL VARIABLE CONTAINS THE AVERAGE OF THE BEST
C                 OVERALL PROPORTION ASH AND THE DESIRED PROPORTION ASH.
C     TARGET      THIS REAL VARIABLE CONTAINS THE TARGET YIELD.
C     DELTA       THIS 1-DIMENSIONAL REAL ARRAY CONTAINS THE INCRE-
C                 MENTAL AMOUNT NEEDED TO CLEAN TO ACHEIVE THE TARGET
C                 BY SPECIFIC GRAVITY FRACTION.
C     ASHDIF      THIS REAL VARIABLE CONTAINS THE DIFFERENCE BETWEEN
C                 AN ACTUAL PERCENT ASH AND THE TARGET PERCENT.
C                 THIS INTEGER VARIABLE CONTAINS THE SEQUENCE NUMBER
C                 OF THE SPECIFIC GRAVITY FRACTION COMING CLOSEST TO
C                 THE DESIRED ASH VALUE.
C     F           THIS REAL VARIABLE CONTAINS THE ADJUSTED VALUE FOR
C                 THE BEST DISTRIBUTION VALUE.
C
      COMMON/WORK/NEXT,LAST,ICV    ,IUNIT,IDATA,ITITLE,ICHAR,IFLW,IASH,
     1 ISUL,IBTU
      COMMON/BLK1/ISFD,ISCC,ISRF,ISMX,SYLD(24),TYLD(24),EFFIC(24),
     1 BTURFC(24),ASHERR(24),SFLT(24),SINK(24),SMISPL(24),SNRGR(24),
     2 SGRAV(24),SPE(24),SIMP(24),SEA(24),SUMRY(24,40)
      COMMON/LOCALV/NS,NG,SCMWT(20),SUMASH(20),YIELD(23),ASH(23),IJX
     1             ,DMIN,ICURVE,NCURVE,NPOINT,IX,IY,J1,SUM1,RATIO,SUM2
```

FIGURE II-5-7.1. (Continued)

```
     2                 ,SUM3,YLDPTS(20),ASHPTS(20),S(20),SGSP1,SGSP2,SGAV
     3                 ,DIFF,IJY,SGSP,X,XYLD,XASH,YLDAV,ASHAV,TARGET
     4                 ,DELTA(20),ASHDIF,JK,F
      DIMENSION FEED(NSIZE,NGRAV,4),CC(NSIZE,NGRAV,4),REF(NSIZE,NGRAV,4)
     1            ,SIZE(23),GRAV(NGRAV),GBOUND(NGRAV),EQUIP(4)
      COMMON WS(6000)
      INTEGER EQUIP,WS
C
C     STEP 1          COMPUTE ONE PLUS THE NUMBER OF SIZE AND SPECIFIC
C                     GRAVITY FRACTIONS.
C
      NS=NSIZE+1
      NG=NGRAV+1
C
C     STEP 2          COMPUTE THE DESIRED VALUES OF YIELD AND PROPORTION
C                     ASH FOR EACH SIZE FRACTION.
C
      DO 5 I=1,NSIZE
      IF(SUMRY(I,ISFD).LT.1.E-6) GOTO 5
      SCMWT(1)=0.
      SCMASH(1)=0.
      WT=0.0
      DO 12 J=1,NGRAV
   12 WT=WT+FEED(I,J,1)
      DO 4 J=2,NG
      J1=J-1
      SCMWT(J)=0.
      SCMASH(J)=0.
      DO 3 K=1,J1
      SCMWT(J)=SCMWT(J)+FEED(I,K,1)
    3 SCMASH(J)=SCMASH(J)+FEED(I,K,IASH)
      SCMASH(J)=SCMASH(J)/SCMWT(J)*100.0
      IF(SCMASH(J).LT.SCMASH(J-1))SCMASH(J)=SCMASH(J-1)
      SCMWT(J)=SCMWT(J)*100.0/WT
    4 CONTINUE
      YIELD(I)=.01*YINTRP(GBOUND,SCMWT,1.50,NG)
      ASH(I)=.01*YINTRP(GBOUND,SCMASH,1.60,NG)
    5 CONTINUE
C
C     STEP 3          INITIALIZE AT ONE THE DUMMY COUNTER TO CONTROL THE
C                     MOVEMENT THROUGH THE SIZE FRACTIONS.
C
      DO 10 I=1,NSIZE
C
C     STEP 4          IF THERE IS MATERIAL IN THE CURRENT SIZE FRACTION,
C                     BRANCH TO STEP 6.
C
      IF(SUMRY(I,ISFD).GE.1.E-6) GOTO 70
C
```

FIGURE II-5-7.1. (Continued)

```
C      STEP 5         THE CURRENT SIZE FRACTION IS EMPTY. ZERO OUT THE
C                     VALUES FOR THE CLEAN AND REFUSE OUTPUT FLOWS, AND
C                     BRANCH TO STEP 18.
C
       DO 69 J=1,NGRAV
       DO 69 K=1,NCHAR
       CC(I,J,K)=0.0
       REF(I,J,K)=0.0
    69 CONTINUE
       GO TO 10
C
C      STEP 6         A DISTRUBUTION CURVE MUST BE DERIVED FOR THE CURRENT
C                     SIZE FRACTION. INITIALIZE THE BASIC VARIABLES NEEDED
C                     TO MOVE THROUGH THE CURVES.
C
    70 IJX=0
       DMIN=100.0
       ICURVE=EQUIP(4)
       NCURVE=WS(ICURVE+2)
       NPOINT=WS(ICURVE+1)
       IX=WS(ICURVE+3)-NPOINT
       IY=WS(ICURVE+4)-NPOINT
C
C      STEP 7         INITIALIZE AT ONE THE DUMMY COUNTER USED TO CONTROL
C                     THE MOVEMENT THROUGH THE INDIVIDUAL RAW GENERALIZED
C                     DISTRIBUTION CURVES.
C
       DO 8 I1=1,NCURVE
C
C      STEP 8         UPDATE THE VARIOUS CONTROL VARIABLES SO THAT THEY
C                     POINT TO THE CURRENT CURVE.
C
       J1=I1
       IX=IX+NPOINT
       IY=IY+NPOINT
C
C      STEP 9         MOVING THROUGH ALL DESIRED SPECIFIC GRAVITIES OF
C                     SEPARATION BEGINNING AT 1.41 AND ENDING AT 1.60 IN
C                     INCREMENTS OF 0.01 DETERMINE THE YIELD AND ASH
C                     WHICH WOULD RESULT FROM EACH OF THE ACTUAL DISTRI-
C                     BUTION CURVES WHICH WOULD BE GENERATED.
C
       DO 7 INDEX=1,20
       SUM1=0.
       SUM2=0.
       SUM3=0.
       S(INDEX)=1.40+.01*FLOAT(INDEX)
       DO 6 J=1,NGRAV
       X=GRAV(J)/S(INDEX)
```

FIGURE II-5-7.1. (Continued)

```
      RATIO=YINTRP(WS(IX),WS(IY),X,NPOINT)
      SUM1=SUM1+FEED(I,J,1)
      SUM2=SUM2+RATIO*FEED(I,J,1)
      SUM3=SUM3+RATIO*FEED(I,J,IASH)
    6 CONTINUE
      YLDPTS(INDEX)=SUM2/SUM1
      ASHPTS(INDEX)=SUM3/SUM2
      IF(INDEX.EQ.1)GO TO 7
      IF(ASHPTS(INDEX).LT.ASHPTS(INDEX-1))ASHPTS(INDEX)=ASHPTS(INDEX-1)
    7 CONTINUE
C
C     STEP 10        COMPUTE THE SEPARATION GRAVITY GIVING THE DESIRED
C                    YIELD AND ASH VALUE. COMPUTE THE AVERAGE OVERALL
C                    SEPARATION GRAVITY AS THEIR AVERAGE AND COMPUTE
C                    THEIR DIFFERENCE. IF THEIR DIFFERENCE EXCEEDS THE
C                    MINIMUM DIFFERENCE COMPUTED THUS FAR, BRANCH TO
C                    STEP 12.
C
      IF((YLDPTS(1).GT.YIELD(I)).OR.(YLDPTS(20).LT.YIELD(I)))GO TO 8
      IF((ASHPTS(1).GT.ASH(I)).OR.(ASHPTS(20).LT.ASH(I)))GO TO 8
      SGSP1=YINTRP(YLDPTS,S,YIELD(I),20)
      SGSP2=YINTRP(ASHPTS,S,ASH(I),20)
      IF(ASHPTS(20)-ASHPTS(1).LE.0.00001)SGSP2=SGSP1
      SGAV=0.5*(SGSP1+SGSP2)
      DIFF=ABS(SGSP2-SGSP1)
      IF(DIFF.GE.DMIN)GO TO 8
C
C     STEP 11        THE CURRENT CURVE MAY BE THE DESIRED CURVE. UPDATE
C                    CONTROL VARIABLES ACCORDINGLY.
C
      IJX=IX
      IJY=IY
      SGSP=SGAV
      DMIN=DIFF
C
C     STEP 12        INCREMENT BY ONE THE DUMMY COUNTER CONTROLLING THE
C                    MOVEMENT THROUGH THE RAW GENERALIZED CURVES. IF ALL
C                    HAVE NOT BEEN PROCESSED, LOOP BACK TO STEP 8.
C
    8 CONTINUE
C
C     STEP 13        COMPUTE THE VALUES FOR THE OUTPUT FLOWS BASED ON
C                    THE BEST OVERALL DISTRIBUTION CURVE FOUND.
C
      SUM1=0.
      SUM2=0.
      SUM3=0.
      DO 11 J=1,NGRAV
      DO 9 K=1,NCHAR
```

FIGURE II-5-7.1. (Continued)

```
      CC(I,J,K)=YIELD(I)*FEED(I,J,K)
      IF(IJY.EQ.0)GO TO 80
      X=GRAV(J)/SGSP
      CC(I,J,K)=YINTRP(WS(IJX),WS(IJY),X,20)*FEED(I,J,K)
   80 REF(I,J,K)=FEED(I,J,K)-CC(I,J,K)
    9 CONTINUE
      SUM1=SUM1+FEED(I,J,1)
      SUM2=SUM2+CC(I,J,1)
      SUM3=SUM3+CC(I,J,IASH)
   11 CONTINUE
C
C     STEP 14       IF NO GOOD CURVE WAS FOUND, BRANCH TO STEP 18.
C
      IF(IJX.EQ.0)GO TO 92
C
C     STEP 15       COMPUTE THE OVERALL VALUES OF YIELD AND ASH DERIVED
C                   FROM THE BEST OVERALL CURVE. IF THE YIELD VALUE
C                   IS SUFFICIENTLY CLOSE TO THE DESIRED VALUE, BRANCH
C                   TO STEP 18.
C
      XYLD=SUM2/SUM1
      XASH=SUM3/SUM2
      IF(ABS(YIELD(I)-XYLD).LE.1.E-4)GO TO 92
C
C     STEP 16       COMPUTE A SPECIFIC GRAVITY FRACTION SPECIFIC ADJUST-
C                   MENT FACTOR TO BE USED TO ADJUST THE DISTRIBUTION
C                   CURVE VALUE IN ORDER TO OBTAIN THE DESIRED VALUES.
C
      YLDAV=.5*(YIELD(I)+XYLD)
      ASHAV=.5*(ASH(I)+XASH)
      TARGET=YLDAV*(ASH(I)-XASH)/(YIELD(I)-XYLD)
      DMIN=100.0
      DO 90 J=1,NGRAV
      DELTA(J)=0.
      ASHDIF=CC(I,J,IASH)/CC(I,J,1)-ASHAV
      DIFF=ABS(TARGET-ASHDIF)
      IF(DIFF.GE.DMIN)GO TO 90
      JK=J
      DMIN=DIFF
   90 CONTINUE
      DELTA(JK)=(YIELD(I)-XYLD)/FEED(I,JK,1)
C
C     STEP 17       ADJUST THE VALUES OF THE OUTPUT FLOWS TO REFLECT
C                   THE ADJUSTMENT NEEDED.
C
      DO 91 J=1,NGRAV
      X=GRAV(J)/SGSP
      F=YINTRP(WS(IJX),WS(IJY),X,NPOINT)+DELTA(J)
      IF(F.GT.1.)F=1.
```

FIGURE II-5-7.1. (Continued)

```
      IF(F.LT.0.)F=0.
      DO 91 K=1,NCHAR
      CC(I,J,K)=F*FEED(I,J,K)
      REF(I,J,K)=FEED(I,J,K)-CC(I,J,K)
   91 CONTINUE
C
C     STEP 18       INCREMENT BY ONE THE DUMMY COUNTER CONTROLLING THE
C                   MOVEMENT THROUGH THE SIZE FRACTIONS, AND IF ALL HAVE
C                   NOT BEEN PROCESSED, LOOP BACK TO STEP 4.
C
   92 CONTINUE
   10 CONTINUE
C
C     STEP 19       END LOCAL PROCESSING.
C
      RETURN
      END
```

FIGURE II-5-7.1. (Continued)

5.8 Subroutine BREAK

A key operation performed by the rotary breaking unit operation for both the coal stream and the rock stream components of the feed involves simulating the actual breakage, selection for breakage, and screening in the drum. These operations are carried out in subroutine BREAK. Figure II-5-8.1 contains the listing and basic documentation for this routine.

```
      SUBROUTINE BREAK(OVER,UNDER,PROD,BR,C,FALL,NSIZE,NGRAV,NCHAR
     1                ,NFALL,HGT,FOVER,FUNDER,HSF)
C
C     TITLE(SIMULATE BREAKING OPERATION)
C
C     SUBROUTINE BREAK IS A SLAVE ROUTINE TO THAT SUBROUTINE WHICH
C     CONTROLS THE SIMULATION OF THE ROTARY BREAKER. THE PARTICULAR
C     PURPOSE OF THIS ROUTINE IS TO ESTIMATE THE EFFECT OF THE ACTUAL
C     BREAKING PROCESS ON THE FLOW OF COAL.
C
C     PARAMETER      DESCRIPTION
C     OVER           THIS 3-DIMENSIONAL REAL ARRAY CONTAINS THE DESCRIP-
C                    TION OF THE FLOW OF COAL IN THE OVERFLOW. WHEN CALLED
C                    THIS FLOW IS THE FEED TO THE BREAKER. NOTE THAT THE
C                    VALUES IN THIS ARRAY ARE STORED IN ADDITIVE FORM WHEN
C                    THIS ROUTINE IS CALLED.
C     UNDER          THIS 3-DIMENSIONAL REAL ARRAY RETURNS THE DESCRIP-
C                    TION OF THE UNDERFLOW STREAM FROM THE BREAKER.
C     PROD           THIS 3-DIMENSIONAL, REAL ARRAY IS USED AS A SCRATTCH
C                    STORAGE ARRAY BY THIS ROUTINE.
C     BR             THIS 2-DIMENSIONAL UPPER TRIANGULAR ARRAY CONTAINS
C                    THE PROPORTION OF COAL IN A GIVEN SIZE FRACTION
C                    WHICH WILL BREAK INTO A SIZE SMALLER THAN OR EQUAL
C                    TO ANOTHER SIZE FRACTION.
C     C              THIS 1-DIMENSIONAL, REAL ARRAY CONTAINS THE PROBAB-
C                    ILITY BY SIZE FRACTION THAT COAL WILL NOT PASS
C                    THROUGH THE HOLES IN THE DRUM OF THE BREAKER.
C     FALL           THIS 1-DIMENSIONAL ARRAY CONTAINS THAT PROPORTION
C                    OF THE COAL TO BE SELECTED FOR BREAKAGE ON EACH FALL.
C     NSIZE          THIS INTEGER VARIABLE DEFINES THE NUMBER OF SIZE
C                    FRACTIONS USED TO CLASSIFY THE FLOWS OF COAL.
C     NGRAV          THIS INTEGER VARIABLE DEFINES THE NUMBER OF SPECIFIC
C                    GRAVITY FRACTIONS USED TO CLASSIFY THE FLOWS OF COAL.
C     NCHAR          THIS INTEGER VARIABLE CONTAINS THE NUMBER OF CHAR-
C                    ACTERISTICS PLUS ONE DEFINED FOR THE FLOWS OF COAL.
C     NFALL          THIS INTEGER VARIABLE CONTAINS THE NUMBER OF FALLS
C                    TO BE TAKEN BY THE COAL.
C     HGT            THIS REAL VARIABLE CONTAINS THE HEIGHT WHICH THE
C                    COAL IS TO DROP ON EACH FALL.
C     FOVER          THIS REAL VARIABLE RETURNS THE TOTAL AMOUNT OF
C                    COAL, BY WEIGHT, IN THE OVERFLOW STREAM.
C     FUNDER         THIS REAL VARIABLE RETURNS THE TOTAL AMOUNT OF
C                    COAL, BY WEIGHT, IN THE UNDERFLOW STREAM.
C     HSF            THIS REAL VARIABLE CONTAINS THE HEIGHT SELECTION
C                    FACTOR FOR MAKING THE BREAKAGE ADJUSTMENT.
C
C     VARIABLE       DESCRIPTION
C     NS             THIS INTEGER VARIABLE CONTAINS ONE MINUS THE NUMBER
C                    OF SIZE FRACTIONS. IT IS THIS NUMBER OF SIZE FRAC-
```

FIGURE II-5-8.1. LISTING FOR SUBROUTINE BREAK

```
C                   TIONS WHICH CONTAIN MATERIAL WHICH HAS THE OPPOR-
C                   TUNITY TO BE BROKEN INTO MATERIAL IN SMALLER FRAC-
C                   TIONS.
C     I             THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                   MOVEMENT THROUGH THE SIZE FRACTIONS.
C     J             THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                   MOVEMENT THROUGH THE SPECIFIC GRAVITY FRACTIONS.
C     K             THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                   MOVEMENT THROUGH THE COAL CHARACTERISTICS.
C     NN            THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                   MOVEMENT THROUGH THE FALLS OF COAL.
C     SEL           THIS REAL VARIABLE CONTAINS THAT PROPORTION OF COAL
C                   SELECTED FOR BREAKAGE ON A GIVEN FALL.
C     L             THIS DUMMY INTEGER VARIABLE IS USED TO SPECIFY THE
C                   SEQUENCE NUMBER OF THAT SIZE FRACTION IMMEDIATELY
C                   SMALLER THAN THE SIZE FRACTION WHOSE MATERIAL IS
C                   CURRENTLY BEING BROKEN.
C     II            THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                   MOVEMENT THROUGH THOSE SIZE FRACTIONS RECEIVING
C                   BROKEN MATERIAL.
C     P             THIS REAL VARIABLE CONTAINS THAT PROPORTION OF
C                   COAL IN A GIVEN SIZE FRACTION TO BE BROKEN INTO
C                   ANOTHER GIVEN SIZE FRACTION.
C
      DIMENSION OVER(NSIZE,NGRAV,NCHAR),UNDER(NSIZE,NGRAV,NCHAR)
     1          ,PROD(NSIZE,NGRAV,NCHAR),BR(23,23),C(NSIZE),FALL(NFALL)
C
C     STEP 1        COMPUTE ONE MINUS THE NUMBER OF SIZE FRACTIONS. IT
C                   IS THIS NUMBER OF FRACTIONS WHICH MIGHT LOSE MATERIAL
C                   BECAUSE OF BREAKAGE.
C
      NS=NSIZE-1
C
C     STEP 2        ZERO OUT THE CURRENT AMOUNT OF MATERIAL IN THE
C                   UNDERFLOW STREAM.
C
      DO 15 I=1,NSIZE
      DO 15 J=1,NGRAV
      DO 10 K=1,NCHAR
      UNDER(I,J,K)=0.0
   10 CONTINUE
   15 CONTINUE
C
C     STEP 3        INITIALIZE AT ONE THE DUMMY COUNTER TO BE USED TO
C                   CONTROL THE MOVEMENT THROUGH THE FALLS OF COAL.
C
      DO 60 NN=1,NFALL
C
C     STEP 4        COMPUTE THE PROPORTION OF COAL WHICH IS ELIGIBLE FOR
```

FIGURE II-5-8.1. (Continued)

```
C                  BREAKAGE ON THE CURRENT FALL.
C
      SEL=FALL(NN)
      IF(HGT.GT.HSF)SEL=SEL*HGT/HSF
C
C     STEP 5       MOVE THAT COAL WHICH IS NOT ELIGLE FOR BREAKAGE
C                  ON THE CURRENT FALL INTO THE SCRATCH STORAGE ARRAY.
C
      DO 20 J=1,NGRAV
      DO 20 K=1,NCHAR
      PROD(NSIZE,J,K)=OVER(NSIZE,J,K)
      DO 20 I=1,NS
      PROD(I,J,K)=OVER(I,J,K)*(1.0-SEL)
   20 CONTINUE
C
C     STEP 6       IF NO COAL IS ELIGIBLE FOR BREAKAGE ON THE CURRENT
C                  FALL( BRANCH TO STEP 8.
C
      IF(SEL.EQ.0.0)GO TO 35
C
C     STEP 7       COMPUTE THE NEW DISTRIBUTION OF COAL WHICH RESULTED
C                  FROM THE BREAKAGE ON THE CURRENT FALL, AND STORE IT
C                  IN THE SCRATCH STORAGE ARRAY.
C
      DO 30 I=1,NS
      L=I
      DO 25 II=L,NSIZE
      P=(BR(I,II)-BR(I,II+1))*SEL
      DO 25 J=1,NGRAV
      DO 25 K=1,NCHAR
      PROD(II,J,K)=PROD(II,J,K)+OVER(I,J,K)*P
   25 CONTINUE
   30 CONTINUE
C
C     STEP 8       AT THIS POINT THE SCRATCH STORAGE ARRAY CONTAINS ALL
C                  OF THE MATERIAL IN THE OVERFLOW AT THE START OF THE
C                  CURRENT FALL IN THE NEW SIZE DISTRIBUTION.  COMPUTE
C                  THE PROPORTION OF THAT MATERIAL WHICH WILL NOT CURR-
C                  ENTLY PASS THROUGH THE HOLES IN THE DRUM. THIS MAT-
C                  ERIAL IS THE OVERFLOW STREAM FOR THE NEXT FALL. THE
C                  MATERIAL WHICH DOES PASS THROUGH IS ADDED INTO THE
C                  UNDERFLOW STREAM.
C
   35 DO 50 I=1,NSIZE
      DO 50 J=1,NGRAV
      DO 50 K=1,NCHAR
      OVER(I,J,K)=C(I)*PROD(I,J,K)
      PROD(I,J,K)=PROD(I,J,K)-OVER(I,J,K)
      UNDER(I,J,K)=UNDER(I,J,K)+PROD(I,J,K)
```

FIGURE II-5-8.1. (Continued)

```
   50 CONTINUE
C
C     STEP 9          INCREMENT THE DUMMY COUNTER CONTROLING THE MOVEMENT
C                     THROUGH THE FALLS BY ONE, AND IF ALL FALLS HAVE NOT
C                     BEEN SIMULATED LOOP BACK TO STEP 4.
C
   60 CONTINUE
C
C     STEP 10         COMPUTE THE WEIGHT OF COAL IN THE OVERFLOW AND
C                     UNDERFLOW STREAMS.
C
      FUNDER=0.0
      FOVER=0.0
      DO 70 I=1,NSIZE
      DO 70 J=1,NGRAV
      FUNDER=FUNDER+UNDER(I,J,1)
      FOVER=FOVER+OVER(I,J,1)
   70 CONTINUE
C
C     STEP 11         END LOCAL PROCESSING.
C
      RETURN
      END
```

FIGURE II-5-8.1. (Continued)

5.9 Subroutine ROTARY

Subroutine ROTARY performs the calculations needed to simulate the rotary breaking unit operation. Figure II-5-9.1 contains the listing and basic documentation for this routine.

Reproduced from
best available copy.

```
      SUBROUTINE ROTARY(FEED,CPROD,RPROD,COVER,ROVER,CUNDR,RUNDR,SIZE
     1                  ,EQUIP,NSIZE,NGRAV,NCHAR,FCPROD,FRPROD,LENGTH
     2                  ,DIAM,HOLE)
C
C     TITLE(SIMULATE ROTARY BREAKER)
C
C     SUBROUTINE ROTARY PERFORMS THE CALCULATIONS NEEDED TO SIMULATE
C     THE UNIT OPERATION OF ROTARY BREAKING.
C
C     PARAMETER     DESCRIPTION
C     FEED          THIS REAL, 3-DIMENSIONAL ARRAY CONTAINS THE DISTRI-
C                   BUTION AND CHARACTERISTICS OF THE FEED TO THE UNIT.
C     CPROD         THIS REAL, 3-DIMENSIONAL ARRAY RETURNS THE DISTRI-
C                   BUTION AND CHARACTERISTICS OF THE UNDERFLOW FROM
C                   THE ROTARY BREAKER.
C     RPROD         THIS REAL, 3-DIMENSIONAL ARRAY RETURNS THE DISTRI-
C                   BUTION AND CHARACTERISTICS OF THE OVERFLOW FROM THE
C                   ROTARY BREAKER.
C     COVER         THIS REAL, 3-DIMENSIONAL ARRAY RETURNS THE DISTRI-
C                   BUTION AND CHARACTERISTICS OF THE COAL COMPONENT OF
C                   THE OVERFLOW FROM THE ROTARY BREAKER.
C     ROVER         THIS REAL, 3-DIMENSIONAL ARRAY RETURNS THE DISTRI-
C                   BUTION AND CHARACTERISTICS OF THE ROCK COMPONENT
C                   OF THE OVERFLOW STREAM FROM THE ROTARY BREAKER.
C     CUNDR         THIS REAL, 3-DIMENSIONAL ARRAY RETURNS THE DISTRI-
C                   BUTION AND CHARACTERISTICS OF THE COAL COMPONENT OF
C                   THE UNDERFLOW FROM THE ROTARY BREAKER.
C     RUNDR         THIS REAL, 3-DIMENSIONAL ARRAY RETURNS THE DISTRI-
C                   BUTION AND CHARACTERISTICS OF THE ROCK COMPONENT OF
C                   THE UNDERFLOW STREAM FROM THE ROTARY BREAKER.
C     SIZE          THIS REAL, 1-DIMENSIONAL ARRAY CONTAINS THE SIZE
C                   DISTRIBUTION DEFINITION FOR THE FLOWS.
C     EQUIP         THIS REAL, 1-DIMENSIONAL ARRAY CONTAINS THE EQUIP-
C                   MENT INFORMATION CONTROL BLOCK.
C     NSIZE         THIS INTEGER VARIABLE DEFINES THE NUMBER OF SIZE
C                   FRACTIONS DEFINING THE FLOWS.
C     NGRAV         THIS INTEGER VARIABLE CONTAINS THE NUMBER OF SPECI-
C                   FIC GRAVITY FRACTIONS DEFINING THE FLOWS.
C     NCHAR         THIS INTEGER VARIABLE CONTAINS ONE PLUS THE NUMBER
C                   OF CHARACTERISTICS DEFINED FOR THE FLOWS.
C     FCPROD        THIS REAL VARIABLE RETURNS THE AMOUNT OF MATERIAL
C                   IN THE UNDERFLOW FROM THE ROTARY BREAKER.
C     FRPROD        THIS REAL VARIABLE RETURNS THE AMOUNT OF MATERIAL
C                   IN THE OVERFLOW FROM THE ROTARY BREAKER.
C     LENGTH        THIS REAL VARIABLE CONTAINS THE LENGTH IN FEET OF THE
C                   DRUM. IT IS THE FIRST DECISION VARIABLE.
C     DIAM          THIS REAL VARIABLE CONTAINS THE DIAMETER IN FEET OF
C                   THE DRUM. IT IS THE SECOND DECISION VARIABLE.
C     HOLE          THIS REAL VARIABLE CONTAINS THE HOLE SIZE IN INCHES.
```

FIGURE II-5-9.1. LISTING FOR SUBROUTINE ROTARY

```
C                   IT IS THE THIRD DECISION VARIABLE.
C
C     EXTERNALS     NOTPTH, EXP, BREAK, SDSIZE
C
C     COMMON        VARIABLES USED
C     //            WS
C     /BLK1/        SYLD,SFLT,SINK,SUMRY,ISCC,ISMX,ISRF
C     /WORK/        IASH
C
C     VARIABLE      DESCRIPTION
C     SMID          THIS 1-DIMENSIONAL REAL ARRAY CONTAINS THE MID-
C                   POINTS OF THE SIZE FRACTIONS.
C     BR            THIS 2-DIMENSIONAL REAL ARRAY CONTAINS THE PROPOR-
C                   TION OF MATERIAL IN THE I-TH SIZE FRACTION THAT WILL
C                   BE IN THE J-TH FRACTION OR BELOW AFTER BREAKAGE.
C     SEL, ISEL     THIS TYPELESS VARIABLE CONTAINS THE BASE ADDRESS
C                   IN WORKING STORAGE OF THE CURVE SET CONTROL BLOCKS
C                   FOR THE BREAKAGE FALL SELECTION FACTORS.
C     NFALL         THIS INTEGER VARIABLE CONTAINS THE NUMBER OF FALLS
C                   TO BE TAKEN IN THE BREAKER.
C     HGT           THIS REAL VARIABLE CONTAINS THE HEIGHT OF EACH FALL
C                   IN THE BREAKER.
C     A             THIS REAL VARIABLE CONTAINS THE EMPIRICAL SCREENING
C                   VALUE FOR THE MATERIAL PASSING THROUGH THE HOLES IN
C                   THE BREAKER.
C     I,II,K        THIS INTEGER DUMMY VARIABLE IS USED TO CONTROL THE
C                   MOVEMENT THROUGH VARIOUS LOCAL LOOPS.
C     J             THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                   MOVEMENT THROUGH THE SPECIFIC GRAVITY FRACTIONS.
C     NS            THIS INTEGER VARIABLE CONTAINS 1 LESS THAN THE NUMBER
C                   OF SIZE FRACTIONS CLASSIFYING THE FLOWSTREAMS.
C     NG            THIS INTEGER VARIABLE CONTAINS ONE LESS THAN THE NUM-
C                   BER OF SPECIFIC GRAVITY FRACTIONS CLASSIFYING THE
C                   FLOWSTREAMS.
C     L             THIS DUMMY INTEGER VARIABLE USED AS A LOWER BOUND
C                   FOR LOOPS PROCESSING THE TRIANGULAR BREAKAGE ARRAY.
C     RATIOB        THIS REAL VARIABLE IS USED TO CONTAIN THE ARGUMENT
C                   FOR THE BREAKAGE CALCULATION. IT CONTAINS THE RATIO
C                   OF THE SIZES.
C     FROCK         THIS REAL VARIABLE CONTAINS THE TOTAL WEIGHT OF
C                   MATERIAL IN THE ROCK COMPONENT OF THE FEED TO THE
C                   ROTARY BREAKER.
C     FCR           THIS REAL VARIABLE CONTAINS THE WEIGHT OF MATERIAL
C                   IN THE OVERFLOW STREAM FROM THE ROCK COMPONENT.
C     FCC           THIS REAL VARIABLE CONTAINS THE TOTAL WEIGHT OF MATER
C                   IAL IN THE UNDERFLOW STREAM FROM THE COAL COMPONENT.
C     FRR           THIS REAL VARIABLE CONTAINS THE TOTAL WEIGHT OF MAT-
C                   ERIAL IN THE OVERFLOW STREAM OF THE ROCK COMPONENT.
C     FRC           THIS REAL VARIABLE CONTAINS THE TOTAL WEIGHT OF MAT-
```

FIGURE II-5-9.1. (Continued)

```
C                      ERIAL FROM THE UNDERFLOW STREAM OF THE ROCK COMPONENT
C      RATIO           THIS DUMMY REAL VARIABLE CONTAINS VARIOUS WEIGHT
C                      RATIOS IN THE CALCULATION OF THE UNIT PERFORMANCE
C                      CHARACTERISTICS.
C      NSC             THIS INTEGER VARIABLE CONTAINS THE ROW NUMBER IN THE
C                      VARIOUS SUMMARY ARRAYS OF THE COMPOSITE VALUES.
C      IS              THIS INTEGER VARIABLE CONTAINS THE ADDRESS IN
C                      WORKING STORAGE OF THE ACTUAL SELECTION FACTORS.
C      TOTAL           THIS REAL VARIABLE IS USED FOR TEMPORARY STORAGE.
C
      DIMENSION FEED(NSIZE,NGRAV,NCHAR),CPROD(NSIZE,NGRAV,NCHAR)
     1          ,RPROD(NSIZE,NGRAV,NCHAR),COVER(NSIZE,NGRAV,NCHAR)
     2          ,ROVER(NSIZE,NGRAV,NCHAR),CUNDR(NSIZE,NGRAV,NCHAR)
     3          ,RUNDR(NSIZE,NGRAV,NCHAR),SIZE(NSIZE),EQUIP(10)
      COMMON/BLK1/ISFD,ISCD,ISRF,ISMX,SYLD(24),TYLD(24),EFFIC(24),
     1 ETUREC(24),ASHERR(24),SFLT(24),SINK(24),SMISPL(24),SNRGR(24),
     2 SGRAV(24),SPE(24),SIMP(24),SEA(24),SUMRY(24,40)
      COMMON/WORK/NEXT,LAST,ICURVE,IUNIT,IDATA,ITITLE,ICHAR,IFLW,IASH,
     1 ISUL,IBTU
      COMMON/LOCALV/SMID(23),SEL,NFALL,HGT,A,NS,NG,L,RATIOB,FROCK,FCR
     1             ,FCC,FRR,FRC,RATIO,NSC,IS,LVFILL(156)
      COMMON WS(6000)
      DIMENSION BR(23,23)
      REAL LENGTH
      INTEGER WS
      EQUIVALENCE (BR(1,1),SUMRY(1,1))
      EQUIVALENCE (SEL,ISEL)
C
C     STEP 1         OBTAIN THE ADDRESS IN WORKING STORAGE OF THE CURVES
C                    DEFINING THE FALL SELECTION VALUES.
C
      SEL=EQUIP(4)
C
C     STEP 2         COMPUTE THE NUMBER AND HEIGHT OF THE FALLS IN THE
C                    DRUM OF THE ROTARY BREAKER.
C
      NFALL=IFIX(LENGTH/EQUIP(6))
      HGT=.75*DIAM
C
C     STEP 3         OBTAIN THE SCREENING EQUATION EXPERIMENTAL CONSTANT
C                    BASED ON THE EQUIPMENT SPECIFICATION AND THE HOLE
C                    SIZE.
C
      A=EQUIP(8)
      IF(HOLE.GT.EQUIP(7))A=EQUIP(9)
C
C     STEP 4         USING SUBROUTINE  N O T P T H  COMPUTE THE PROPOR-
C                    TION OF MATERIAL BY SIZE FRACTION THAT WILL NOT
C                    PASS THROUGH THE HOLES IN THE DRUM ON A SINGLE PASS.
```

FIGURE II-5-9.1. (Continued)

```
C
      CALL NOTPTH(SIZE,SYLD,HOLE,A,NSIZE)
      DO 10 I=1,NSIZE
      IF(SYLD(I).LT.1.0E-4)SYLD(I)=0.0
   10 CONTINUE
C
C     STEP 5        COMPUTE THE MIDPOINTS OF THE SIZE FRACTIONS.
C
      DO 15 I=1,NSIZE
      SMID(I)=(SIZE(I)+SIZE(I+1))/2.0
   15 CONTINUE
C
C     STEP 6        USING THE GENERAL BREAKAGE EQUATION COMPUTE THE
C                   VALUES IN THE BREAKAGE MATRIX.
C
      NS=NSIZE-1
      DO 25 I=1,NS
      BR(I,I)=1.0
      L=I+1
      DO 20 II=L,NSIZE
      RATIOB=SIZE(II)/SMID(I)
      BR(I,II)=(1.0-EXP(-RATIOB))/0.63214
   20 CONTINUE
      BR(I,NSIZE+1)=0.0
   25 CONTINUE
C
C     STEP 7        LIBERATE THE ROCK AND COAL COMPONENT FLOWS FROM
C                   THE FEED TO THE BREAKER. EXPRESS THE DESCRIPTIONS
C                   OF THE COMPONENTS IN ADDITIVE FORM.
C
      FROCK=0.
      NG=NGRAV-1
      DO 35 I=1,NSIZE
      DO 30 J=1,NGRAV
      DO 30 K=1,NCHAR
      COVER(I,J,K)=FEED(I,J,K)
      ROVER(I,J,K)=0.0
   30 CONTINUE
      ROVER(I,NGRAV,1)=COVER(I,NGRAV,IASH)-COVER(I,NG,IASH)
      IF(ROVER(I,NGRAV,1).LT.0.0)ROVER(I,NGRAV,1)=0.0
      COVER(I,NGRAV,1)=COVER(I,NGRAV,1)-ROVER(I,NGRAV,1)
      ROVER(I,NGRAV,IASH)=ROVER(I,NGRAV,1)
      COVER(I,NGRAV,IASH)=COVER(I,NGRAV,IASH)-ROVER(I,NGRAV,IASH)
   35 CONTINUE
C
C     STEP 8        USING SUBROUTINE  B R E A K  SIMULATE THE PASSAGE
C                   OF THE COAL COMPONENT FLOW THROUGH THE BREAKER.
C
      IS=WS(ISEL+3)
```

FIGURE II-5-9.1. (Continued)

```
      CALL BREAK(COVER,CUNDR,CPROD,BR,SYLD,WS(IS),NSIZE,NGRAV,NCHAR
     1 ,NFALL,HGT,FCR,FCC,EQUIP(10))
C
C     STEP 9        USING SUBROUTINE  B R E A K  SIMULATE THE PASSAGE
C                   OF THE ROCK COMPONENT FLOW THROUGH THE BREAKER.
C
      IS=WS(ISEL+9)
      CALL BREAK(ROVER,RUNDR,RPROD,BR,SYLD,WS(IS),NSIZE,NGRAV,NCHAR
     1 ,NFALL,HGT,FRR,FRC,EQUIP(10))
C
C     STEP 10       BLEND THE COAL UNDERFLOW COMPONENT AND THE ROCK
C                   UNDERFLOW COMPONENT TO FORM THE COMPOSITE UNDERFLOW
C                   FROM THE BREAKER.
C
      DO 36 I=1,NSIZE
      DO 36 J=1,NGRAV
      DO 36 K=1,NCHAR
      CPROD(I,J,K)=CUNDR(I,J,K)+RUNDR(I,J,K)
   36 CONTINUE
C
C     STEP 11       BLEND THE COAL OVERFLOW COMPONENT AND THE ROCK
C                   OVERFLOW COMPONENT TO FORM THE COMPOSITE OVERFLOW
C                   FROM THE BREAKER.
C
      DO 37 I=1,NSIZE
      DO 37 J=1,NGRAV
      DO 37 K=1,NCHAR
      RPROD(I,J,K)=COVER(I,J,K)+ROVER(I,J,K)
   37 CONTINUE
C
C     STEP 12       USING SUBROUTINE  S D S I Z E  SUMMARIZE THE DESCRIP-
C                   OF THE FLOWSTREAMS BY SIZE FRACTION AND COMPOSITE.
C
      CALL SDSIZE(RPROD,SUMRY(1,ISRF),NSIZE,NGRAV,NCHAR,FRPROD)
      CALL SDSIZE(CPROD,SUMRY(1,ISCC),NSIZE,NGRAV,NCHAR,FCPROD)
C
C     STEP 13       COMPUTE THE ROW NUMBER IN THE SUMMARY ARRAYS OF THE
C                   ROW CONTAINING THE COMPOSITE VALUES.
C
      NSC=NSIZE+1
C
C     STEP 14       COMPUTE THE UNIT YIELD FROM THE PROBABILITY OF
C                   NOT PASSING THROUGH THE HOLES.
C
      DO 40 I=1,NSIZE
      SYLD(I)=100.0*(1.0-SYLD(I))
   40 CONTINUE
      SYLD(NSC)=100.0*FCC/(FCC+FCR)
C
```

FIGURE II-5-9.1. (Continued)

```
C     STEP 15        USING SUBROUTINE  S D S I Z E  SUMMARIZE BY SIZE
C                    FRACTION AND COMPOSITE THE CHARACTERISTICS OF THE
C                    COAL COMPONENT IN THE OVERFLOW STREAM.
C
      CALL SDSIZE(COVER,SUMRY(1,ISMX),NSIZE,NGRAV,NCHAR,TOTAL)
      RATIO=100.0*FCR/(FCR+FRR)
      DO 45 I=1,NSC
      SFLT(I)=0.0
      IF(SUMRY(I,ISRF).GT.0.0) SFLT(I)=SUMRY(I,ISMX)/SUMRY(I,ISRF)*RATIO
   45 CONTINUE
C
C     STEP 16        USING SUBROUTINE  S D S I Z E  CALCULATE THE DISTRI-
C                    BUTION OF THE ROCK COMPONENT IN THE UNDERFLOW STREAM.
C
      CALL SDSIZE(RUNDR,SUMRY(1,ISMX),NSIZE,NGRAV,NCHAR,TOTAL)
      RATIO=100.0*FRC/(FRC+FCC)
      DO 50 I=1,NSC
      SINK(I)=0.0
      IF(SUMRY(I,ISCC).GT.0.0) SINK(I)=SUMRY(I,ISMX)/SUMRY(I,ISCC)*RATIO
   50 CONTINUE
C
C     STEP 17        END LOCAL PROCESSING.
C
      RETURN
      END
```

FIGURE II-5-9.1. (Continued)

5.10 Subroutine CRUSH

The crushing unit operation is implemented in subroutine CRUSH. Figure II-5-10.1 contains the listing and basic documentation.

```
      SUBROUTINE CRUSH(FEED,PROD,FED,SIZE,EQUIP,NSIZE,NGRAV,NCHAR,SO)
C
C     TITLE(SIMULATE CRUSHER)
C
C     SUBROUTINE CRUSH IS USED TO CONTROL THE SIMULATION OF THE CRUSHING
C     UNIT OPERATION.
C
C     PARAMETER     DESCRIPTION
C     FEED          THIS REAL, 3-DIMENSIONAL ARRAY CONTAINS THE DISTRI-
C                   BUTION AND CHARACTERISTICS OF THE FEED TO THE UNIT.
C     PROD          THIS REAL, 3-DIMENSIONAL ARRAY RETURNS THE DISTRI-
C                   BUTION AND CHARACTERISTICS OF THE PRODUCT FROM THE
C                   CRUSHER.
C     FED           THIS REAL, 3-DIMENSIONAL ARRAY CONTAINS THE DISTRI-
C                   BUTION AND CHARACTERISTICS OF THE MATERIAL IN THE
C                   CRUSHER IMMEDIATELY AFTER THE MATERIAL LARGER THAN
C                   THE CRUSHING ZONE HAS BEEN BROKEN.
C     SIZE          THIS REAL, 1-DIMENSIONAL ARRAY CONTAINS THE SIZE
C                   DISTRIBUTION DEFINITION.
C     EQUIP         THIS REAL, 1-DIMENSIONAL ARRAY CONTAINS THE EQUIP-
C                   MENT INFORMATION CONTROL BLOCK.
C     NSIZE         THIS INTEGER VARIABLE DEFINES THE NUMBER OF SIZE
C                   FRACTIONS DEFINING THE FLOWS.
C     NGRAV         THIS INTEGER VARIABLE CONTAINS THE NUMBER OF SPECIFIC
C                   GRAVITY FRACTIONS DEFINING THE FLOWS.
C     NCHAR         THIS INTEGER VARIABLE CONTAINS THE NUMBER OF CHAR-
C                   ACTERISTICS PLUS ONE DEFINED FOR THE FLOWSTREAMS.
C     SO            THIS REAL VARIABLE CONTAINS THE VALUE OF THE CRUSHER
C                   SETTING.
C
C     EXTERNALS     INTS, SDSIZE
C
C     COMMON        VARIABLES USED
C     //            WS
C     /BLK1/        SUMRY
C
C     VARIABLE      DESCRIPTION
C     BR            THIS REAL, 2-DIMENSIONAL ARRAY IS USED TO CONTAIN
C                   THE PROPORTION OF MATERIAL IN EACH SIZE FRACTION
C                   WHICH WILL BE BROKEN INTO PIECES OF THE OTHER SIZE
C                   FRACTIONS.
C     CURVE,ICURVE THIS TYPELESS VARIABLE CONTAINS THE ADDRESS IN
C                   WORKING STORAGE OF THE CURVE SET DEFINING THE EQUIP-
C                   MENT TYPE SPECIFIC BREAKAGE DISTRIBUTION.
C     NPOINT        THIS INTEGER VARIABLE CONTAINS THE NUMBER OF POINTS
C                   IN THE EQUIPMENT BREAKAGE DISTRIBUTION.
C     IX            THIS INTEGER VARIABLE CONTAINS THE ADDRESS IN WORKING
C                   STORAGE OF THE INDEPENDENT VALUES ASSOCIATED WITH
C                   THE EQUIPMENT SPECIFIC BREAKAGE DISTRIBUTION.
```

FIGURE II-5-10.1. LISTING FOR SUBROUTINE CRUSH

```
C      IY                 THIS INTEGER VARIABLE CONTAINS THE ADDRESS IN WORKING
C                         STORAGE OF THE DEPENDENT VALUES ASSOCIATED WITH THE
C                         EQUIPMENT BREAKAGE DISTRIBUTION.
C      I                  THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                         MOVEMENT THROUGH LOCAL LOOPS.
C      J                  THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                         MOVEMENT THROUGH LOCAL LOOPS.
C      SMID               THIS 1-DIMENSIONAL REAL ARRAY CONTAINS THE MIDPOINTS
C                         OF THE SIZE FRACTIONS.
C      NNS                THIS INTEGER VARIABLE CONTAINS ONE PLUS THE NUMBER
C                         OF SIZE FRACTIONS.
C      M1                 THIS INTEGER VARIABLE CONTAINS THE SEQUENCE NUMBER
C                         OF THAT SIZE FRACTION CONTAINING THE MAXIMUM OF THE
C                         CRUSHING ZONE. THIS SIZE FRACTION WILL BE REFERRED
C                         TO AS THE CRUSHING FRACTION.
C      M                  THIS INTEGER VARIABLE CONTAINS THE SEQUENCE NUMBER OF
C                         THE FRACTION IMMEDIATELY LARGER THAN THE CRUSHING
C                         FRACTION.
C      MM                 THIS INTEGER VARIABLE CONTAINS THE SEQUENCE NUMBER OF
C                         THE SIZE FRACTION IMMEDIATELY SMALLER THAN THE
C                         CRUSHING FRACTION.
C      NS                 THIS INTEGER VARIABLE CONTAINS ONE MINUS THE NUMBER
C                         OF SIZE FRACTIONS.
C      L                  THIS INTEGER DUMMY VARIABLE IS USED AS THE LOWER
C                         LOOP LIMIT IN PROCESSING THE TRIANGULAR BREAKAGE
C                         ARRAY.
C      PR                 THIS REAL 1-DIMENSIONAL ARRAY CONTAINS THE PROPO-
C                         TION OF MATERIAL IN A GIVEN SIZE FRACTION THAT WILL
C                         BREAK INTO MATERIAL EQUAL OR SMALLER THAN THE OTHER
C                         SIZE FRACTIONS.
C      II                 THIS INTEGER DUMMY VARIABLE IS USED TO CONTROL
C                         THE PROCESSING OF INNER LOOPS.
C      RATIOB             THIS REAL VARIABLE EQUALS THE ARGUMENT TO THE
C                         BREAKAGE DISTRIBUTIONS. IT CONTAINS THE RATIO OF
C                         THE RELEVANT SIZE FRACTION MIDPOINTS.
C      Z                  THIS REAL VARIABLE CONTAINS THE PROPORTION OF
C                         MATERIAL IN A GIVEN FRACTION WHICH WILL BE BROKEN
C                         TO MATERIAL IN ANOTHER GIVEN FRACTION.
C      P                  THIS REAL VARIABLE CONTAINS THE AMOUNT OF MATERIAL
C                         MOVING FROM ONE FRACTION TO ANOTHER.
C      FRAC1              THIS DUMMY REAL VARIABLE CONTAINS THE FRACTION OF
C                         MATERIAL IN A GIVEN SIZE FRACTION TO WHICH A GIVEN
C                         CALCULATION APPLIES.
C      FRAC2              THIS REAL DUMMY VARIABLE CONTAINS THE REMAINING
C                         FRACTION OF A GIVEN SIZE FRACTION.
C      SMID1              THIS DUMMY REAL VARIABLE CONTAINS A MIDPOINT BETWEEN
C                         A SIZE FRACTION BOUNDARY AND SOME CRUSHER SIZE
C                         BOUNDARY.
C      SMID2              THIS REAL DUMMY VARIABLE CONTAINS THE MIDPOINT BE-
```

FIGURE II-5-10.1. (Continued)

```
C                   TWEEN A CRUSHER SIZE BOUNDARY AND A FLOWSTREAM SIZE
C                   BOUNDARY. .
C     LL            THIS DUMMY INTEGER VARIABLE IS USED IN THE CALCU-
C                   LATION OF DIRECT RATIOS FROM CUMULATIVE RATIOS.
C     SEL1          THIS REAL VARIABLE CONTAINS THE RATIO OF MATERIAL
C                   SELECTED FOR SOME CALCULATION.
C     SEL2          THIS REAL VARIABLE CONTAINS THE RATIO OF MATERIAL
C                   SELECTED FOR SOME CALCULATION.
C     SEL           THIS REAL VARIABLE CONTAINS THE OVERALL RATIO
C                   OF MATERIAL SELECTED.
C     RATIOS        THIS REAL VARIABLE CONTAINS THE RATIO OF A SIZE AND
C                   THE CRUSHER SETTING FOR USE IN THE SCREENING EQUATION.
C     VAL           THIS REAL VARIABLE CONTAINS THE AMOUNT OF MATERIAL
C                   BEING MOVED AFTER BREAKAGE.
C     K             THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                   MOVEMENT THROUGH FLOW CHARACTERISTICS.
C
      DIMENSION FEED(NSIZE,NGRAV,NCHAR),PROD(NSIZE,NGRAV,NCHAR)
     1          ,FED(NSIZE,NGRAV,NCHAR),SIZE(NSIZE),EQUIP(8),BR(23,23)
      COMMON/LOCALV/PR(23),SMID(23),ICURVE,IX,IY,NPOINT,NMS,M1,M,MM,NS
     1             ,L,RATIOB,Z,P,FRAC1,FRAC2,SMID1,SMID2,LL,SEL1,SEL2
     2             ,SEL,RATIOS,VAL,LVFILL(126)
      COMMON/BLK1/ISFD,ISCC,ISRF,ISMX,SYLD(24),TYLD(24),EFFIC(24),
     1 BTUREC(24),ASHERP(24),SFLT(24),SINK(24),SMISPL(24),SMRGR(24),
     2 SGRAV(24),SPE(24),SIMP(24),SEA(24),SUMRY(24,40)
      COMMON WS(6000)
      INTEGER WS
      EQUIVALENCE (CURVE,ICURVE)
      EQUIVALENCE (BR(1,1),SUMRY(1,1))
C
C     STEP 1        OBTAIN THE ADDRESSES IN WORKING STORAGE OF THE INDE-
C                   PENDENT AND DEPENDENT VALUES ASSOCIATED WITH THE
C                   BREAKAGE DISTRIBUTION FOR THE EQUIPMENT TYPE.
C
      CURVE=EQUIP(4)
      NPOINT=WS(ICURVE+1)
      IX=WS(ICURVE+3)
      IY=WS(ICURVE+4)
C
C     STEP 2        INITIALIZE AT ZERO THE ARRAY WHICH WILL CONTAIN THE
C                   PROPORTION OF MATERIAL IN GIVEN SIZE FRACTIONS WHICH
C                   WILL BE CRUSHED INTO OTHER FRACTIONS.
C
      DO 51 I=1,NSIZE
      DO 51 J=1,NSIZE
      BR(I,J)=0.0
   51 CONTINUE
C
C     STEP 3        COMPUTE THE MIDPOINTS OF THE SIZE FRACTIONS.
```

FIGURE II-5-10.1. (Continued)

```
C
      DO 10 I=1,NSIZE
   10 SMID(I)=(SIZE(I)+SIZE(I+1))/2.0
C
C     STEP 4        COMPUTE THE TOTAL NUMBER OF VALUES DEFINING THE SIZE
C                   DISTRIBUTION. THIS IS ONE PLUS THE NUMBER OF SIZE
C                   FRACTIONS.
C
      NNS=NSIZE+1
C
C     STEP 5        COMPUTE THE SEQUENCE NUMBER OF THE CRUSHING FRACTION.
C
      DO 3 I=1,NNS
      IF(SIZE(I)-EQUIP(7)*SO)4,3,3
    4 M1=I-1
      GO TO 8
    3 CONTINUE
C
C     STEP 6        INITIALIZE AT ZERO THE FLOW DESCRIPTION ARRAYS
C                   WHICH WILL CONTAIN THE VALUES FOR THE CRUSHED MATER-
C                   IAL.
C
    8 DO 21 I=1,NSIZE
      DO 21 J=1,NGRAV
      DO 21 K=1,NCHAR
      FED(I,J,K)=FEED(I,J,K)
   21 PROD(I,J,K)=0.0
C
C     STEP 7        COMPUTE THE SEQUENCE NUMBERS OF THE SIZE FRACTIONS
C                   IMMEDIATELY ABOVE AND IMMEDIATELY BELOW THE CRUSHING
C                   FRACTION. THEN IF ALL FRACTIONS ARE BELOW THE CRUSH-
C                   ING FRACTION BRANCH TO STEP 19.
C
      M=M1-1
      MM=M1+1
      IF(M1.EQ.0)GO TO 90
C
C     STEP 8        COMPUTE THE SEQUENCE NUMBER OF THE NEXT TO SMALLEST
C                   SIZE FRACTION. THIS IS THE LARGEST FRACTION WHICH
C                   CAN LOSE MATERIAL AS A RESULT OF CRUSHING.
C
      NS=NSIZE-1
C
C     STEP 9        IF THERE IS NO MATERIAL IN SIZE FRACTIONS SMALLER
C                   THAN THE CRUSHING FRACTION, BRANCH TO STEP 15.
C
      IF(M1.EQ.1)GO TO 50
C
C     STEP 10       INITIALIZE AT ONE THE DUMMY COUNTER TO CONTROL THE
```

FIGURE II-5-10.1. (Continued)

```
                 MOVEMENT THROUGH THE SIZE FRACTIONS WHICH ARE LARGER
                 THAN THE CRUSHING FRACTION.

      DO 24 I=1,M

      STEP 11    USING SUBROUTINE  I N T S  AND THE EQUIPMENT TYPE
                 SPECIFIC BREAKAGE DISTRIBUTION COMPUTE THE CUMULATIVE
                 PROPORTION OF MATERIAL THAT WILL BREAK INTO SMALLER
                 FRACTIONS.

      L=I+1
      PR(I)=1.0
      DO 9 II=L,NSIZE
      RATIOB=SMID(II)/SMID(I)
      CALL INTS(WS(IX),WS(IY),RATIOB,PR(II),NPOINT)
    9 CONTINUE
      PR(NSIZE)=0.0

      STEP 12    PHYSICALLY SIMULATE THE BREAKAGE OF MATERIAL FROM
                 FRACTIONS LARGER THAN THE CRUSHING FRACTION INTO
                 FRACTIONS NOT SMALLER THAN THE CRUSHING FRACTION.

      DO 23 II=L,M1
      Z=PR(II-1)-PR(II)
      DO 23 J=1,NGRAV
      DO 23 K=1,NCHAR
      FED(II,J,K)=FED(II,J,K)+FED(I,J,K)*Z
   23 CONTINUE

      STEP 13    COMPUTE THE PROPORTION OF MATERIAL IN THE CURRENT
                 FRACTION WHICH WILL BREAK INTO FRACTIONS SMALLER
                 THAN THE CRUSHING FRACTION.

      DO 25 II=MM,NSIZE
      LL=II-1
      P=PR(LL)-PR(II)
      BR(I,II)=BR(I,II)+P
   25 CONTINUE

      STEP 14    INCREMENT BY ONE THE DUMMY COUNTER CONTROLLING THE
                 MOVEMENT THROUGH THE FRACTIONS LARGER THAN THE CRUSH-
                 ING FRACTION, AND IF NOT ALL HAVE BEEN PROCESSED,
                 LOOP BACK TO STEP 11.

   24 CONTINUE

      STEP 15    COMPUTE THAT PROPORTION OF MATERIAL IN THE CRUSHING
                 FRACTION WHICH IS LARGER THAN THE CRUSHING ZONE MAX-
                 IMUM AND COMPUTE THE CUMULATIVE PROPORTION OF THAT
```

FIGURE II-5-10.1. (Continued)

Reproduced from best available copy.

```
C                       MATERIAL WHICH WILL BREAK INTO SMALLER FRACTIONS
C                       USING SUBROUTINE  I N T S  AND THE EQUIPMENT TYPE
C                       SPECIFIC BREAKAGE DISTRIBUTION.
C
   50 PP(M1)=1.0
      FRAC1=(SIZE(M1)-EQUIP(7)*SO)/(SIZE(M1)-SIZE(MM))
      FRAC2=1.0-FRAC1
      SMID1=(SIZE(M1)+EQUIP(7)*SO)/2.0
      SMID2=(EQUIP(7)*SO+SIZE(MM))/2.0
      DO 66 II=MM,NSIZE
      RATIOB=SMID(II)/SMID1
      CALL INTS(WS(IX),WS(IY),RATIOB,PR(II),NPOINT)
   66 CONTINUE
      PR(NSIZE)=0.0
C
C     STEP 16       ADD THE DIRECT PROPORTION OF MATERIAL BREAKAGE FROM
C                   THE PARTIAL FRACTION INTO THE BREAKAGE ARRAY.
C
      DO 26 II=MM,NSIZE
      LL=II-1
      P=PR(LL)-PR(II)
      BR(M1,II)=BR(M1,II)+P*FRAC1
   26 CONTINUE
C
C     STEP 17       USING THE GENERAL BREAKAGE DISTRIBUTION COMPUTE
C                   THE CUMULATIVE DISTRIBUTION OF MATERIAL IN THAT PART
C                   OF THE CRUSHING FRACTION SMALLER THAN THE MAXIMUM OF
C                   THE CRUSHING ZONE INTO SMALLER FRACTIONS.
C
      DO 61 II=MM,NSIZE
      RATIOB=SMID(II)/SMID2
      PR(II)=(1.0-EXP(-RATIOB))/0.63214
   61 CONTINUE
      PR(NSIZE)=0.0
      PR(M1)=EQUIP(6)
C
C     STEP 18       ADD THE CONTRIBUTION OF THE RATIOS COMPUTED IN STEP
C                   17 INTO THE BREAKAGE ARRAY.
C
      DO 62 II=MM,NSIZE
      PR(II)=PR(II)*EQUIP(6)
      LL=II-1
      P=PR(LL)-PR(II)
      BR(M1,II)=BR(M1,II)+P*FRAC2
   62 CONTINUE
      BR(M1,M1)=BR(M1,M1)+FRAC2*(1.0-EQUIP(6))
C
C     STEP 19       INITIALIZE THE DUMMY COUNTER USED TO CONTROL THE
C                   MOVEMENT THROUGH THE SIZE FRACTIONS BELOW THE CRUSH-
```

FIGURE II-5-10.1. (Continued)

```
C                   ING FRACTION SO THAT IT POINTS TO THE FIRST SUCH
C                   SIZE FRACTION.
C
   90 IF(MM.GT.NS) GOTO 41
      DO 49 I=MM,NS
C
C     STEP 20       USING THE GENERAL BREAKAGE FUNCTION COMPUTE THE
C                   CUMULATIVE BREAKAGE VALUES FOR THE CURRENT FRACTION.
C
      L=I+1
      DO 43 II=L,NSIZE
      RATIOB=SMID(II)/SMID(I)
      PR(II)=(1.0-EXP(-RATIOB))/0.63214
   43 CONTINUE
      PR(NSIZE)=0.0
C
C     STEP 21       COMPUTE THAT PROPOTION OF THE MATERIAL IN THE
C                   CURRENT FRACTION WHICH WILL BE SELECTED FOR BREAK-
C                   AGE. THE ASSUPTION HERE IS THAT SOME MATERIAL WILL
C                   SLIP THROUGH THE CRUSHER. NOTE THAT THE STANDARD
C                   SCREENING FORMULATION IS USED HERE.
C
      SEL1=0.0
      SEL2=0.0
      FRAC1=1.0
      FRAC2=1.0
      IF(SIZE(I)-SO)44,44,45
   45 SEL1=EQUIP(6)
      IF(SIZE(I+1)-SO)76,46,46
   76 FRAC1=(SIZE(I)-SO)/(SIZE(I)-SIZE(I+1))
      FRAC2=1.0-FRAC1
      SMID2=(SO+SIZE(I+1))/2.0
      RATIOS=SMID2/SO
      GO TO 74
   44 RATIOS=SMID(I)/SO
   74 SEL2=EXP(EQUIP(8)*(RATIOS-1.0))
   46 DO 47 II=L,NSIZE
      SEL=SEL1*FRAC1+SEL2*FRAC2*EQUIP(6)
   47 PR(II)=PR(II)*SEL
      PR(I)=SEL
C
C     STEP 22       ADD THE BREAKAGE CONTRIBUTION COMPUTED ABOVE INTO THE
C                   BREAKAGE ARRAY.
C
      L=I+1
      DO 54 II=L,NSIZE
      LL=II-1
      P=PR(LL)-PR(II)
      BR(I,II)=BR(I,II)+P
```

FIGURE II-5-10.1. (Continued)

```
   54 CONTINUE
C
C     STEP 23         INCREMENT BY ONE THE DUMMY COUNTER CONTROLLING THE
C                     MOVEMENT THROUGH THE SIZE FRACTIONS BELOW THE CRUSH-
C                     ING FRACTION, AND IF NOT ALL HAVE BEEN PROCESSED,
C                     LOOP BACK TO STEP 20.
C
   49 CONTINUE
C
C     STEP 24         MOVE THE PROPORTION OF MATERIAL NOT SELECTED FOR
C                     BREAKAGE IN EACH FRACTION BELOW THE CRUSHING FRAC-
C                     TION INTO THE DIAGONAL ENTRIES OF THE BREAKAGE ARRAY.
C
   41 PR(NSIZE)=0.0
      DO 40 II=MM,NSIZE
      BR(II,II)=BR(II,II)+(1.0-PR(II))
   40 CONTINUE
C
C     STEP 25         ACTUALLY SIMULATE THE APPLICATION OF THE BREAKAGE
C                     COMPUTED ABOVE TO THE RESULT OF THE MATERIAL OBTAINED
C                     AFTER THE BREAKAGE OF THE FRACTIONS ABOVE THE CRUSH-
C                     ING FRACTION.
C
   85 DO 52 I=1,NSIZE
      DO 52 II=I,NSIZE
      DO 53 J=1,NGRAV
      DO 53 K=1,NCHAR
      PROD(II,J,K)=PROD(II,J,K)+FED(I,J,K)*BR(I,II)
   53 CONTINUE
   52 CONTINUE
C
C     STEP 26         USING SUBROUTINE  S D S I Z E  SUMMARIZE THE VALUES
C                     FOR THE PRODUCT FLOW BY SIZE FRACTION AND COMPOSITE.
C
      CALL SDSIZE(PROD,SUMRY(1,ISCC),NSIZE,NGRAV,NCHAR,VAL)
C
C     STEP 27         END LOCAL PROCESSING
C
      RETURN
      END
```

FIGURE II-5-10.1. (Continued)

6.0 CHARACTER MANIPULATION

Any computer program that is to be directly controllable by or informative to a human user must be able to accept and produce coded records. Coded records are constructed of sequences of characters. Thus, any FORTRAN program that is to communicate with a human user must be able to deal with characters. The only obvious mechanisms through which characters can be dealt with in FORTRAN are the coded READ and WRITE statements with associated FORMAT. This chapter will show that these mechanisms cannot by themselves be used to establish coded communications of the quality desired. The reason for this is that their use requires that the programmer know the FORMAT of a record to be read or written at the time he writes the program.

The process of dealing with characters in a computer program is referred to by the term character manipulation. Generally, the following four capabilities are encompassed under this term:

(1) The ability to read coded records and to decompose them into sequences of characters

(2) The ability to construct coded records from sequences of characters and to write them

(3) The ability to convert subsequence of characters into internal binary representations - fixed point, floating point, or literal

(4) The ability to convert internal binary representations into subsequences of characters.

Within FORTRAN, character manipulation is performed by the system routines which support execution. The programmer does not have access to them; he can only direct their action via formatted READ and WRITE statements. This approach gives the programmer very little flexibility. He needs some techniques to provide additional control over the character manipulation capabilities.

The purpose of this chapter is to present these techniques as they are implemented in CPSM4. First, some problems are presented which can only be solved with great difficulty using standard approaches. Then, the approach taken in CPSM4 is described, first generally, and then in terms of the actual utility routines being used. It is important to remember in reading the discussion in this chapter that CPSM4 is run on many different computers, which created a major design problem. Thus, the character manipulation techniques chosen had to be programmable in a machine-independent form.

6.1 Displaying Size Information

As discussed extensively in various parts of this document, all size information is entered into the program in decimal inch form. This absolute, though admittedly arbitrary rule, simplifies the user's task. He can always be certain what units to use. From the standpoint of the output, however, there is no reason why the same absolute convention need be followed. It is a fairly standard convention to display small size measurements in mesh rather than in inches, and to display larger values in fractional rather than decimal form. Ideally, then, program CPSM4, when it generates an output report, should follow the same output convention. The program does follow this convention.

Unfortunately, the convention of showing size values in either mesh or frational form, when possible, cannot be implemented using a simple FORMAT statement, since there is no FORMAT specification letter which performs the desired converion - i.e., which generates the character sequence 1 1/2 when presented with the floating point value of 1.5 or which generates the character sequence 28M when presented with the floating point value of 0.0232. The program itself then needs routines which can perform the desired conversions.

The approach taken in the original version of the program was the exact inverse of the one taken in the present version. In that version the user entered the size fraction classification scheme in fractional or mesh form (without the suffix M). The program then converted these into inch form and retained the display forms for use in its output tables. There are two objections to this approach - a serious one and a trivial one.

The trivial objection is that the convention of saving the input characters for later display purposes requires additional redundant core storage. The objection is trivial since the additional storage needed is slight.

The serious objection is that having the program convert the input to inches can cause interpretation problems. An actual example will illustrate this point. In a recent application of the program, one of the size fraction break points was specified as 9 mesh, equal to approximately 2 millimeters or 0.08 inches. The table lookup in the routine which converted the input to inches had entries for 8 and for 10 mesh, but not for 9 mesh. This is because 9 is not a frequently used mesh classification. The user of the program, in this case the author who is unfamiliar with the fine points of size classification schemes, did not realize that 9 would not be recognized as a mesh. The convention in the program at that time was to assume that any input size definition which was not recognizable as a mesh should be assumed to be in inches; therefore, the internal size value assigned was 9 inches and not 0.08 inches. The output, however, always displayed the 9, which was still believed by the author to be a recognized mesh size. The program results were, however, bizarre since it treated those fractions which were 0.08 inches as being 9 inches. Considerable time was spent in locating the problem.

No such problems can occur with the present convention. The program will always use the size range definitions as entered by the user. If those definitions contain even fractions or meshes, then they will be displayed as such. If the user makes a mistake, then that mistake will appear explicitly in the output so that he can see it.

The actual routine used to generate the appropriate size display, PRSIZE, will be discussed after another problem has been presented and after some machinery has been introduced.

6.2 Displaying Unit Descriptors

The other general problem requiring some extension of the standard FORTRAN FORMAT approach has to do with the unit descriptors. CPSM4 produces three reports which include unit descriptors:

(1) The units input report
(2) The unit performance reports
(3) The summary of data for units reports.

Examples of all these reports can be found in part I-2.1 which presents the output for the handbook example. In the units input report, the unit descriptor appears beginning in print column 51 surrounded by parentheses. In the unit performance reports, the descriptor appears centered under the title. Finally, in the summary data for units report, the descriptor appears beginning in print column 25 again surrounded by parentheses.

In order to place these descriptors on these reports in the manner described above, the programmer has two options. He can hardwire the descriptors into the program and, thus, give the user no control over them; or he can implement some sort of character manipulation routines which will allow him to count the characters in the descriptors, place them at will in the output line, and concatenate them with other characters.

The first alternative was chosen in the original version of program CPSM4. In that version the user had no control whatsoever over the equipment inventory to be available. There were precisely 22 equipment types from which the user could choose. Each equipment type had a hardwired set of curves, identifying code, and descriptor associated with it. Each unit in a particular plant configuration was described via the generic equipment type descriptor. This approach is obviously not appropriate for this version in which the user has broad control over the equipment types and in which individual units are assigned their own descriptors.

The actual routine used to enter the unit descriptors into the various report types is subroutine VESSEL. This routine is documented in Part 2 of Chapter 8.

6.3 Internal Character Display Codes

The previous two parts of this chapter discuss needs of program CPSM4 which require it to have some minimal character manipulation capabilities. The key point to be developed in this part is that if the processes of conversion to and from internal representations can be separated from the process of recognizing and producing sequences of characters, then FORTRAN provides a straightforward capability for doing character manipulation - that of integer arithmetic. The key notion upon which this point rests is that of the internal character display codes. From a methodological standpoint, the approach to be taken here is based on an elemental result from computational linguistics. As Hopcroft and Ullman(5) put it, if a language V* has an alphabet V containing p symbols then: "We can think of the sentences of V* as numbers represented in base p".

The traditional approach to the supposed inability to control character manipulation explicitly via the format approach in FORTRAN is to augment the FORTRAN subroutine and function library with a series of utilities that are written in assembly language and that can perform the needed operations - the most basic of which are masking, shifting, and Boolean combination. In fact, most manufacturers supply such utilities as part of their FORTRAN product set. These utilities are then available to the programmer either as function references or via the CALL statement. This approach has two major drawbacks. In the first place, any program using it is not machine independent. Each time the program is to be run on a different type of machine, the character manipulation utilities have to be rewritten. The second drawback is that the implementation of the character manipulation capabilities strictly via function and subroutine references is clumsy at best. At times, extremely detailed access is needed both to the input character stream and to the output character stream. This access is much easier to achieve if the FORTRAN routines which require it can have direct contact with the character stream. This point will not be developed further here, but several examples of it will be seen in later parts of this chapter.

A final introductory comment before moving into a discussion of the approach to be taken here has to do with the inconsistency of the internal representations of the characters on the various machines. These representations are referred to as the internal character display codes. The observation to be made is that even if a program has access to the individual characters, it is still not necessarily machine independent. Just as with FORMAT statements, the very representation of the character varies from machine to machine.

Assume that there are two utility routines available which have the ability to return the ith character in the input stream. One returns the character left-justified blank-filled to the right, and the other returns it right-justified zero-filled to the left. These are the two forms in which individual characters are typically stored. Suppose now that it is necessary to determine whether or not some character in the input was some particular character - say an A. The statement below in (1)

```
          IF(ICHAR.EQ.1HA)                                        (1)
```

could be used if the character is being returned in ICHAR left-justified, blank-filled to the right; while one of the statements in (2)

```
          IBM 370 - IF(ICHAR.EQ.193)
          UNIVAC - IF(ICHAR.EQ.10)                                (2)
          CDC 6400 - IF(ICHAR.EQ.1)
```

could be used if the character is being returned in ICHAR right-justified, zero filled to the left. The problem with the left-justified, blank-filled to the right approach is that even though it would function properly on any machine, the IF statement is not ANSI FORTRAN. H-type constants, referred to as the literal forms of characters, may appear only in DATA statements and in FORMAT statements in ANSI FORTRAN. The problem with the right-justified, zero-filled to the left approach is even worse. This form of the character simply constitutes the internal display code value in integer form for the character. Integer constants are perfectly acceptable in ANSI FORTRAN IF statements; thus, there is no syntax problem. The problem is semantic. The characters do not have the same internal display codes on different machines. The UNIVAC IF statement would be true on the UNIVAC if and only if the character were an A; while it would be true on the CDC 6400 only if it were a J. There are no standards for what the internal display codes should be for characters.

Using the characters as right-justified, zero-filled to the left values, then, is not feasible if the program is to be usable in a machine-independent context. The alternative would appear to be some version of the left-justified, blank-filled to the right approach modified to avoid the strictly syntactic problem that literal constants are restricted as to where they may appear. This approach is very attractive for another reason. It is very easy to write a utility in ANSI FORTRAN which returns characters left-justified, blank-filled to the right. This is true because a character read under an A1 format specification will be of this form.

The obvious modification of the first approach is to initialize the value of the A literal constant in a variable in a DATA statement. The character could then be compared to this initialized variable. The following shows how this would be written:

```
DATA IA/1HA/
                                                    (3)
IF(ICHAR.EQ.IA)
```

This IF statement, with associated DATA statements, is syntactically well-formed ANSI FORTRAN and will produce the same branch on every computer.

In general, the above logic could be expanded slightly in a manner that might initialize an entire array containing all the characters of interest. If at some point a routine needed to know if some input character represented the ith character of interest, then it could compare that input character to the ith word in the previously initialized list. This might look something like:

```
DIMENSION LET(26)
DATA LET(1),LET(2),LET(3),.../1HA,1HB,1HC,...       (4)
IF(ICHAR.EQ.LET(I))
```

The crucial characteristic of this approach is that the value of some given unknown character could be established by determining its sequence number in the list of possible characters. For each unique character recognizable by the program, there would be one and only one unique sequence number in the list of characters. The approach taken here is based on this observation.

The conclusion, upon which the approach taken here to character manipulation is based, is that the FORTRAN language provides no first order techniques for referencing characters directly if the programs written in that language are to be compiled on different machines. The only method available is to compare the characters to pre-established lists which contain the values for those characters. It is important to note that though the representations established by literal constants in DATA statements vary from machine to machine, the programming statements needed to establish them are the same on every machine. What is important is not what the DATA statement establishes per se, but rather the relationship between what is established and what is received when the characters are transmitted. In particular, the internal representation on any given machine for any given character, say A, when established via the notation 1HA in a DATA statement is identical to the representation received for that character when it is read via an A1 format specification or when it is written via that specification. It does not matter what the actual representations are; it only matters that the above relationship holds for every character on every machine.

Up until this point in the discussion, the general problem of character manipulation has been restricted to the problem of character recognition. The previous discussion has shown how characters can be recognized via IF statements - i.e., how programs can be written which contain sections of code whose execution is contingent on the occurrence of particular characters in particular positions. This ability to branch contingent on the presence of particular characters is not necessarily sufficient to produce an efficient capability to convert strings of characters into internal representations and visa versa. Some stronger capability is needed.

Earlier in this section it was observed that if characters were returned right-justified, zero-filled to the left, then the resulting internal display code values could simply be treated as integer constants. On any particular machine, the attractiveness of this result is that these internal display code values could be used computationally. It is this fact that is used by assembly language routines that do character manipulation. The same operation could be performed in FORTRAN for particular machines if the FORTRAN routines had access to the internal display code values in their integer forms.

As an example, Figure II-6-3.1 shows the internal display code values for the CDC 6400 computer in decimal, integer form for the numeric characters. Assume a function called IRJZF is available, which returns the internal display code value associated with a character stored right-justified, zero filled to the left for the CDC 6400. Using this routine it would be straightforward to write an ANSI FORTRAN routine which converts numeric character strings to their integer equivalents. Figure II-6-3.2 shows a function which performs this operation.

The basic trick used in the above function, IVAL, is that, for the CDC 6400 computer, the actual integer value expressed by a numeric character equals the internal display code value of that character minus 27. Once the expressed values of the characters, stored in the array ICHAR, can be computed for the individual digits in the number, the expressed value of the sequence of characters can be computed directly by doing base ten arithmetic. The basic logic shown in IVAL is exactly the logic used within that CDC FORTRAN system routine which processes I-format specifications (except for error checks and alike).

As pointed out earlier, however, even if there were a function IRJZF available for every machine, function IVAL would still not produce the proper results on computers other than CDC. Function IVAL uses a trick based on the internal display code values for characters on CDC computers. These values are not the same on other computers. IVAL, if executed on an IBM, DEC, or UNIVAC compuer, would produce hash.

What is needed is a technique for producing some set of integer values of the characters on particular machines. In other words, a method is needed to generate a set of machine-independent internal display code values. Whatever tricks were used to convert sequences of these display values into internal representations would then be valid on a machine-independent basis as well.

Character	Internal Display Code	Character	Internal Display Code
0	27	5	32
1	28	6	33
2	29	7	34
3	30	8	35
4	31	9	36

FIGURE II-6-3.1. CDC NUMERIC INTERNAL DISPLAY CODES

```
      FUNCTION IVAL(ICHAR,NCHAR)
      DIMENSION ICHAR(NCHAR)
      IVAL=0
      DO 10 I=1,NCHAR
      IVAL=IVAL*10+IRJZF(ICHAR(I))-27
   10 CONTINUE
      RETURN
      END
```

FIGURE II-6-3.2. SIMPLE NUMERIC CONVERSION ROUTINE

Fortunately, it is very easy to generate a set of machine-independent internal display code values relative to a given set of acceptable characters. It was observed earlier in this section that if at some point a routine needed to know if some input character represented the ith character of interest, then it could compare that input character to the ith word in a previously established list of characters. This previously established list would contain the entire character set of interest. In other words, each character would be represented by one and only one position in the established list. The sequence number of the position in the list matching the character would be a unique integer value corresponding to each character and would not be dependent on the particular machine executing the program. This set of sequence numbers is precisely the set of machine-independent display code values needed.

Figure II-6-3.3 shows a version of function IRJLZ. This is a referenced in function IVAL that returns a machine-independent internal display code value for the alphanumeric characters which coincidentally match the actual CDC internal display code values for those same characters. This version of IRJLZ is written in ANSI FORTRAN, with the exception of the DATA statement which is abbreviated for the sake of simplicity. If this version of IRJLZ were used in conjunction with function IVAL as shown in Figure II-6-3.2, then IVAL would produce the proper results on all machines.

```
      FUNCTION IRJLZ(ICHAR)
      DIMENSION L(36)
      DATA L(1),...,L(26),L(27),...,L(36)/1HA,...,1HZ,1H0,...1H9/
      DO 10 I=1,36
      IF(ICHAR.EQ.L(I))GOTO 15
   10 CONTINUE
      I=0
   15 IRJLZ=I
      RETURN
      END
```

FIGURE II-6-3.3. FUNCTION IRJLZ

The actual approach to internal character display codes, and thus to character manipulation, taken in program CPSM4, can now be presented based upon the previous discussion. The next part of this chapter describes the common block /CHRSET/ which contains the equivalent informaton to that contained in the array L in function IRJLZ shown in Figure II-6-3.3.

6.4 The /CHRSET/ Common Block

The /CHRSET/ common block contains a single integer array LETTER which is dimensioned at 47. This array is initialized as shown in Figure II-6-4.1 in the BLOCK DATA subroutine accompanying program CPSM4. The user might note that this rather clumsy form of the DATA statement is the only one accepted by ANSI FORTRAN. The abbreviated forms of the DATA statement using subscripts or implied subscripts are not allowed. This is the most common violation of ANSI FORTRAN syntax that occurs in programs supposedly written in 'standard FORTRAN'.

```
DATA LETTER(1),LETTER(2),LETTER(3),LETTER(4)/1HA,1HB,1HC,1HD/
DATA LETTER(5),LETTER(6),LETTER(7),LETTER(8)/1HE,1HF,1HG,1HH/
DATA LETTER(9),LETTER(10),LETTER(11),LETTER(12)/1HI,1HJ,1HK,1HL/
DATA LETTER(13),LETTER(14),LETTER(15),LETTER(16)/1HM,1HN,1HO,1HP/
DATA LETTER(17),LETTER(18),LETTER(19),LETTER(20)/1HQ,1HR,1HS,1HT/
DATA LETTER(21),LETTER(22),LETTER(23),LETTER(24)/1HU,1HV,1HW,1HX/
DATA LETTER(25),LETTER(26),LETTER(27),LETTER(28)/1HY,1HZ,1H0,1H1/
DATA LETTER(29),LETTER(30),LETTER(31),LETTER(32)/1H2,1H3,1H4,1H5/
DATA LETTER(33),LETTER(34),LETTER(35),LETTER(36)/1H6,1H7,1H8,1H9/
DATA LETTER(37),LETTER(38),LETTER(39),LETTER(40)/1H+,1H-,1H*,1H//
DATA LETTER(41),LETTER(42),LETTER(43),LETTER(44)/1H(,1H),1H$,1H=/
DATA LETTER(45),LETTER(46),LETTER(47)/1H ,1H,,1H./
```

FIGURE II-6-4.1. INITIALIZATION OF LETTER ARRAY

The array LETTER, then, contains all of the characters which will be recognized by the program in descriptors. This group was chosen since it represents the standard FORTRAN character set and since these are available on all machines. As will be seen later, any character which is not in the above set will be treated as though it were a blank.

Returning now to the basic methodological result from computational linguistics presented in the previous part, it was observed that the sentences in language V* over an alphabet V, containing p symbols, can be represented as numbers base p. All descriptors entering the program can be thought of as sentences of the language LETTER* over the alphabet LETTER. Therefore, all descriptors can be represented as numbers base 47, which is the number of symbols in LETTER. To allow for additional characters that may be necessary, the descriptors are actually represented as base 64 numbers.

Given the above, each individual character, then, must represent a single numeric digit base 64. This implies that each must have a base 10 equivalent value of between 0 and 63. These single digit values are the internal display code values used within the program. They are equal to the sequence number of the character in the LETTER array in the /CHRSET/ common block. Figure II-6-4.2 shows these internal display code values.

Character	Internal display code	Character	Internal display code	Character	Internal display code
A	1	Q	17	6	33
B	2	R	18	7	34
C	3	S	19	8	35
D	4	T	20	9	36
E	5	U	21	+	37
F	6	V	22	-	38
G	7	W	23	*	39
H	8	X	24	/	40
I	9	Y	25	(	41
J	10	Z	26	)	42
K	11	0	27	$	43
L	12	1	28	=	44
M	13	2	29	blank	45
N	14	3	30	,	46
O	15	4	31	.	47
P	16	5	32		

FIGURE II-6-4.2. INTERNAL DISPLAY CODES FOR CPSM4.

6.5 The /CODERC/ Common Block

Before moving into a discussion of the actual character manipulation routines themselves, three additional common blocks must be presented. The first common block, /CODERC/ contains the raw coded record being processed, interpreted, or constructed by the manipulation routine. The characters themselves are either represented in their literal form, left-justified blank-filled to the right, or in the CPSM4 internal display code form depending on the context.

The block itself contains three quantities - ICOL, LCOL, and LINE. ICOL is a simple integer variable that typically indicates the relative starting address in the LINE array at which processing should begin. The LCOL integer variable indicates the relative ending address for processing. The particular uses of the variables is obvious in the routines manipulating them. The LINE array is a one-dimensional integer array containing the unpacked characters.

6.6 The /FILES/ Common Block

The /FILES/ common block contains the unit numbers of the file from which coded input records are to be read, the file to which coded output records are to be sent, and the file to be used for off-line storage. These values are contained in the integer variables READER, PRINTR, and OFFLIN, respectively. Figure II-6-6.1 shows the initialization of these variables as performed in the BLOCK DATA subroutine. Thus, cards are read from TAPE5, written to TAPE6, and no off-line records are written in the current version.

```
DATA READER/5/
DATA PRINTR/6/
DATA OFFLIN/0/
```

FIGURE II-6-6.1. INITIALIZATION OF /FILES/ VARIABLES

6.7 The /CONST/ Common Block

The final common block to be discussed before beginning the actual discussion of the character manipulation routines themselves is the /CONST/ common block. This block contains values of constants needed by the character manipulation routines. Figure II-6-7.1 shows the initialization of the variables in this common block as performed in the BLOCK DATA subroutine. The real one-dimensional array TENS is used to determine the number of decimal places required for a real numeric value. The integer variable ICOM is used by the character manipulation routine CNVTIE as the space between commas in integer values.

```
DATA TENS(1),TENS(2),TENS(3),TENS(4)/10.,100.,1000.,10000./
DATA TENS(5),TENS(6),TENS(7)/100000.,1000000.,10000000./
DATA TENS(8),TENS(9)/100000000.,1000000000./
DATA ICOM /3/
```

FIGURE II-6-7.1. INITIALIZATION OF /CONST/ VARIABLES.

6.8 Subroutine CNVTIE

The fundamental output conversion routine within program CPSM4 is subroutine CNVTIE. This routine converts packed character sequence numbers, fixed point numeric values and floating point numeric values into characters for display. The listing and basic documentation of CNVTIE given in Figure II-6-8.1 are self-explanatory given the discussion in the early part of this chapter.

```
      SUBROUTINE CNVTIE(VALUE,FIELD,ITYPE)
C
C     TITLE(CONVERT TIE VALUE INTO CHARACTERS)
C
C     SUBROUTINE CNVTIE IS USED TO CONVERT TIE VALUES INTO SEQUENCES
C     OF CHARACTER SEQUENCE NUMBERS FOR THE CONSTRUCTION OF CODED RE-
C     CORDS. THE MOST TYPICAL USE OF THESE RECORDS IS FOR DISPLAY.
C
C     PARAMETER     DESCRIPTION
C     VALUE         THIS TYPELESS VARIABLE CONTAINS THE TIE VALUE TO
C                   BE CONVERTED TO CHARACTER SEQUENCE NUMBERS.
C     ITYPE         THIS INTEGER VARIABLE SPECIFIES THE TYPE OF TIE
C                   REPRESENTED BY THE VALUE.
C                   CODE MEANING
C                   -2   THE VALUE CONTAINS PACKED CHARACTER SEQUENCE
C                        NUMBERS.
C                   -1   THE VALUE IS NUMERIC STORED IN FIXED POINT
C                        FORM.
C                   +N   THE VALUE IS NUMERIC STORED IN FLOATING POINT
C                        FORM. N IS BETWEEN 0 AND 9, AND SPECIFIES THE
C                        NUMBER OF DECIMAL PLACES TO BE SHOWN.
C     FIELD         THIS INTEGER VARIABLE DESCRIBES THE SIZE OF THE
C                   FIELD INTO WHICH THE CHARACTER SEQUENCE NUMBERS
C                   ARE TO BE PLACED AND THE ORIENTATION OF THE CHAR-
C                   ACTERS WITHIN THAT FIELD.
C                   CODE MEANING
C                   -N   LEFT-JUSTIFY, BLANK TO RIGHT
C                    0   CONCATENATE
C                   +N   RIGHT JUSTIFY, BLANK FILL TO LEFT.
C
C     VARIABLE      DESCRIPTION
C     I             THIS INTEGER VARIABLE IS A DO-LOOP COUNTER.
C     IADC          THIS INTEGER VARIABLE IS THE OFF-SET COUNTER FOR
C                   NUMERIC VALUES.
C     IVAL,VAL      THIS TYPELESS VARIABLE CONTAINS THE VALUE FOR
C                   NUMERIC DATA.
C     J             THIS INTEGER VARIABLE CONTAINS THE SEQUENCE NUMBER OF
C                   THE CHARACTER BEING PROCESSED.
C     ICC           THIS INTEGER VARIABLE CONTAINS THE SEQUENCE NUMBER
C                   OF THE NEXT SEPARATOR CHARACTER TO BE INSERTED. IT
C                   EITHER INDICATES A DECIMAL POINT (47) OR IT INDI-
C                   CATES A COMMA (46).
C     ICM           THIS INTEGER VARIABLE INDICATES THE NUMBER OF
C                   CHARACTER POSITIONS REMAINING BEFORE ANOTHER SEP-
C                   ARATOR CHARACTER IS TO BE INSERTED.
C     IC            THIS INTEGER VARIABLE SPECIFIES THE ADDRESS OF THE
C                   HIGH-ORDER, LEFT-MOST POSITION IN THE FIELD.
C     LC            THIS INTEGER VARIABLE SPECIFIES THE ADDRESS OF THE
C                   LOW-ORDER, RIGHT-MOST POSITION AVAILABLE FOR THE
```

FIGURE II-6-8.1. LISTING FOR SUBROUTINE CNVTIE

```
C                   FIELD.
C      IDIS, DIS    THIS TYPELESS VARIABLE CONTAINS THE ACTUAL DISPLAY
C                   VALUE.
C      MOD          THIS INTEGER VARIABLE SPECIFIES THE MODULUS USED
C                   TO CONVERT THE DIGITS TO CHARACTER SEQUENCE NUMBERS.
C      IWC          THIS INTEGER VARIABLE SPECIFIES THE ACTUAL NUMBER
C                   OF POSITIONS ALLOCATED FOR THE CHARACTERS.
C      IDIG         THIS INTEGER VARIABLE IS USED AS TEMPORARY STORAGE
C                   FOR THE DISPLAY VALUE AS IT IS BEING MANIPULATED.
C      POW          THIS REAL VARIABLE CONTAINS THE VALUE OF TEN RAISED
C                   TO THE NUMBER OF DECIMAL PLACES FOR THE CONVERSION
C                   OF FLOATING POINT REPRESENTATIONS.
C
C      COMMON       VARIABLES USED
C      /CODERC/     LINE,NCHAR
C      /CHRSET/     LETTER.
C      /CONST/      TENS,ICOM
C
C      EXTERNALS    IABS, IFIX
C
       COMMON/CODERC/NCHAR,LCOL,LINE(150)
       COMMON/CONST/TENS(9),ICOM
       COMMON/CHRSET/LETTER(47)
       INTEGER VALUE,FIELD
       EQUIVALENCE (DIS,IDIS)
       EQUIVALENCE (IVAL,VAL)
       DATA ICC/46/
C
C      STEP 1        COMPUTE THE HIGH-ORDER FIELD POSITION AND IF THE
C                    FIELD HAS A FIXED SIZE INCREMENT THE NUMBER OF
C                    CHARACTERS IN THE CURRENT RECORD ACCORDINGLY. NOTE
C                    THE USE OF THE  F O R T R A N  FUNCTION  I A B S.
C
       IC=NCHAR+1
       NCHAR=NCHAR+IABS(FIELD)
C
C      STEP 2        USING THE  F O R T R A N  FUNCTION  I A B S  SET
C                    THE DISPLAY VALUE EQUAL TO THE ABSOLUTE VALUE OF
C                    THE TIE VALUE, AND BRANCH ACCORDING TO ITS TYPE.
C                    BRANCH     TYPE OF VALUE
C                    STEP 3     CODED
C                    STEP 5     FLOATING POINT
C                    STEP 7     FIXED POINT
C
       IDIS=IABS(VALUE)
       IF(ITYPE+1)10,30,20
C
C      STEP 3        CODED CHARACTERS ARE BEING GENERATED. IF THE VALUE
C                    CONTAINS NONE, BRANCH TO STEP 25.
```

FIGURE II-6-8.1. (Continued)

```
C
   10 IF(IDIS)15,160,15
C
C     STEP 4         SET THE MODULUS, OFFSET, AND SPACING TO PROCESS A
C                    CODED VALUE AND BRANCH TO STEP 9.
C
   15 MOD=64
      IADD=0
      ICM=200
      GO TO 40
C
C     STEP 5         A FLOATING POINT VALUE IS BEING PROCESSED. COMPUTE
C                    THE FIXED POINT DISPLAY VALUE AND IF NO DECIMAL
C                    PLACES ARE REQUESTED, BRANCH TO STEP 7.
C
   20 POW=1.0
      IF(ITYPE.GT.0)POW=TENS(ITYPE)
      IDIS=IFIX(DIS*POW+0.5)
      IF(ITYPE)25,30,25
C
C     STEP 6         A DECIMAL POINT WILL BE NEEDED AS A SEPARATOR. SET
C                    THE SEPARATOR VALUE AND SPACING ACCORDINGLY AND
C                    BRANCH TO STEP 8.
C
   25 ICM=ITYPE
      ICC=47
      GO TO 35
C
C     STEP 7         A NUMERIC VALUE WITH NO DECIMAL PLACES IS BEING
C                    PROCESSED. SET THE SEPARATOR VALUE TO INDICATE A
C                    COMMA (46) AND SET THE SPACING AS INDICATED.
C
   30 ICM=ICOM
      ICC=46
C
C     STEP 8         A NUMERIC VALUE IS BEING PROCESSED. SET THE MODU-
C                    LUS AND OFF-SET ACCORDINGLY.
C
   35 MOD=10
      IADD=27
C
C     STEP 9         IF THE VALUE IS BEING RIGHT JUSTIFIED, BRANCH TO
C                    STEP 16.
C
   40 IF(FIELD)45,45,105
C
C     STEP 10        THE VALUE IS BEING LEFT-JUSTIFIED. THE NUMBER OF
C                    POSITIONS REQUIRED MUST BE COMPUTED. BEGIN BY DET-
C                    ERMINING THE NUMBER OF DIGITS IN THE VALUE.
```

FIGURE II-6-8.1. (Continued)

```
C
   45 IWD=0
      IDIG=IDIS
   50 IDIG=IDIG/MOD
      IWD=IWD+1
      IF(IDIG)50,55,50
C
C     STEP 11        THE NUMBER OF DIGITS HAS BEEN COMPUTED. IF THE
C                    VALUE IS CODED, BRANCH TO STEP 13.
C
   55 IF(IADD)60,85,60
C
C     STEP 12        THE VALUE IS NUMERIC. COMPUTE THE NUMBER OF ADD-
C                    ITIONAL POSITIONS REQUIRED BY THE SIGN, THE COMMA
C                    SEPARATORS, AND THE DECIMAL POINT AS RELEVANT.
C
   60 IF(ITYPE)65,65,70
   65 IWD=IWD+(IWD-1)/ICOM
      GO TO 75
   70 CONTINUE
      IDIG=IDIS/MOD
      IF((IDIS-IDIG*MOD).NE.0) GOTO 71
      IDIS=IDIG
      ICM=ICM-1
      GOTO 70
   71 IWD=IWD+(IWD-ITYPE-1)/ICOM+1
      IVAL=VALUE
      IF(IFIX(VAL).EQ.0) IWD=IWD+1
   75 IF(VALUE)80,85,85
   80 IWD=IWD+1
C
C     STEP 13        THE NUMBER OF POSITIONS REQUIRED BY THE VALUE HAS
C                    BEEN COMPUTED. IF THE VALUE IS TO BE CONCATENATED,
C                    BRANCH TO STEP 15.
C
   85 IF(FIELD)90,100,100
C
C     STEP 14        THE VALUE IS BEING LEFT-JUSTIFIED IN A FIXED FIELD
C                    IF THE REQUIRED NUMBER OF POSITIONS EXCEEDS THE
C                    FIELD WIDTH, SET THEM EQUAL TO THE FIELD WIDTH,
C                    THEN BRANCH TO STEP 17.
C
   90 LC=IC-FIELD-1
      DO 91 I=IC,LC
   91 LINE(I)=45
      IF(IWD+FIELD) 110,110,95
   95 IWD=-FIELD
      GO TO 110
C
```

FIGURE II-6-8.1. (Continued)

```
C     STEP 15     THE VALUE IS BEING CONCATENATED. ADD THE NUMBER OF
C                 POSITIONS REQUIRED BY IT TO THE NUMBER OF CHAR-
C                 ACTERS CURRENTLY IN THE CODED RECORD, AND BRANCH
C                 TO STEP 17.
C
 100  NCHAR=NCHAR+IWD
      GO TO 110
C
C     STEP 16     THE VALUE IS TO BE RIGHT-JUSTIFIED. SET THE NUMBER
C                 OF POSITIONS AVAILABLE EQUAL TO THE FIELD WIDTH.
C
 105  IWD=FIELD
      LC=IC+IWD-1
      DO 106 I=IC,LC
 106  LINE(I)=45
C
C     STEP 17     SET THE INITIAL ADDRESS OF THE LOW ORDER POSITION
C                 EQUAL TO THE ADDRESS OF THE HIGH ORDER POSITION
C                 PLUS THE POSITIONS ALLOCATED.
C
 110  LC=IC+IWD
C
C     STEP 18     THE NEXT CHARACTER SEQUENCE NUMBER MAY NOW BE
C                 PLACED IN THE FIELD. COMPUTE THE NEW LOW ORDER
C                 POSITION. IF IT IS ABOVE THE HIGH-ORDER POSITION,
C                 BRANCH TO STEP 24.
C
 115  LC=LC-1
      IF(LC-IC)155,120,120
C
C     STEP 19     IF A SEPARATOR IS TO BE INSERTED, DO SO, RESET THE
C                 SPACING WIDTH AND CHARACTER VALUE, AND LOOP BACK
C                 TO STEP 18. IF NOT, CONTINUE TO STEP 20.
C
 120  IF(ICM)125,125,130
 125  LINE(LC)=ICC
      ICC=46
      ICM=ICOM
      GO TO 115
C
C     STEP 20     DECREMENT THE SPACING COUNTER, PLACE THE CHARAC-
C                 TER SEQUENCE NUMBER OF THE LOW ORDER DIGIT IN THE
C                 VALUE INTO THE CURRENT LOW ORDER POSITION, AND UP-
C                 DATE THE VALUE.
C
 130  ICM=ICM-1
      IDIG=IDIS/MOD
      LINE(LC)=IDIS-IDIG*MOD+IADD
      IDIS=IDIG
```

FIGURE II-6-8.1. (Continued)

```
C
C     STEP 21        IF THE VALUE CONTAINS DIGITS, OR IF THE DIGIT JUST
C                    POSITIONED IS TO THE RIGHT OF A DECIMAL POINT,
C                    LOOP BACK TO STEP 13.
C
      IF(IDIS)115,135,115
  135 IF(ICC-47)140,115,140
C
C     STEP 22        IF THE TIE VALUE WAS POSITIVE, BRANCH TO STEP 25.
C
  140 IF(VALUE)145,160,160
C
C     STEP 23        THE VALUE WAS NEGATIVE. IF THERE IS ANOTHER POS-
C                    ITION AVAILABLE, INSERT A MINUS SIGN AND BRANCH TO
C                    STEP 25. ELSE, CONTINUE TO STEP 24.
C
  145 LC=LC-1
      IF(LC-IC)155,150,150
  150 LINE(LC)=38
      GO TO 160
C
C     STEP 24        THE VALUE HAS OVERFLOWED THE FIELD. INSERT AN AS-
C                    TERIX IN THE HIGH ORDER POSITION.
C
  155 LINE(IC)=39
C
C     STEP 25        THE VALUE HAS BEEN CONVERTED. RESET THE COMMA
C                    SPACING VARIABLE TO 3 AND END LOCAL PROCESSING.
C
  160 ICOM=3
  999 CONTINUE
      DO 170 I=IC,NCHAR
      J=LINE(I)
  170 LINE(I)=LETTER(J)
      RETURN
      END
```

FIGURE II-6-8.1. (Continued)

6.9 Subroutine PRSIZE

Sufficient machinery has now been introduced to deal directly with the problem introduced in part 6.1 of this chapter. Summarizing, all size information should be displayed in mesh form if it represents an even mesh size or in fractional form if it represents an even fraction (eighths). Otherwise, it should be shown in decimal inches with up to five decimal places. Not mentioned before, the size displays should be left-justified with concatenation to the right, if they are to appear in table row descriptors; and they are to be right-justified if they appear within column headings. Subroutine PRSIZE is the core routine used to implement this set of capabilities. It makes use of subroutine CNVTIE, which was described earlier.

Figure II-6-9.1 shows the mesh size displays produced by subroutine PRSIZE along with their equivalents in decimal inches and their internal display codes shown in base 10. Equivalently, Figure II-6-9.2 shows this same information for the even fractions. The values shown in the last two columns are initialized in the subroutine in a DATA statement. Some explanation of the internal display codes is probably needed.

Let D be a function which specifies the base 10 internal display code value for a given character or sequence of characters. The values of D for single characters are known and are given in Figure II-6-4.2. Now the value of a sequence of characters $C_k,\ldots,C_o$ can be computed from the formula shown in (1)

$$D(C_k,\ldots,C_o) = \sum_{i=k}^{o} D(C_i)\cdot 64^i \tag{1}$$

This equation, of course, is the standard definition of the base 10 value of a base 64 number. To see how this calculation works, consider the computation shown in (2) of the internal display code value of the display 3/8.

$$\begin{aligned} D(3/8) &= D(3)\text{x}64\text{x}64+D(/)\text{x}64+D(8) \\ &= 30\text{x}4096+40\text{x}64+35\text{x}1 \\ &= 122880+2560+35 \\ &= 125475 \end{aligned} \tag{2}$$

This is the internal display code value shown for 3/8 in Figure II-6-9.2.

Mesh display	Decimal inches	Internal display
4M	0.185	1997
6M	0.131	2125
8M	0.093	2253
10M	0.065	116429
14M	0.046	116685
20M	0.0328	120525
28M	0.0232	121037
35M	0.0164	124941
48M	0.0116	129229
65M	0.0082	137229
100M	0.0058	7452365
150M	0.0041	7472845
200M	0.0029	7714509
270M	0.0021	7743181
325M	0.0017	7985165
400M	0.0015	8238797

FIGURE II-6-9.1. MESH SIZES

Fractional display	Decimal inches	Internal display code
7/8	0.875	141859
3/4	0.75	125471
5/8	0.625	133667
1/2	0.5	117277
3/8	0.375	125475
1/4	0.25	117279
1/8	0.125	117283

FIGURE II-6-9.2. FRACTIONAL SIZES

At first glance, the above calculation looks very imposing; however, they actually go quite quickly on a desk calculator. A simpler form to compute is shown in (3).

$$D(3/8) = (D(3)\times64+D(/))\times64+D(8) \qquad (3)$$

Given the initialization of the internal display codes, the remainder of the routine is straightforward as described in the basic documentation. Figure II-6-9.3 contains the flowchart for PRSIZE; while Figure II-6-9.4 contains the listing and basic documentation.

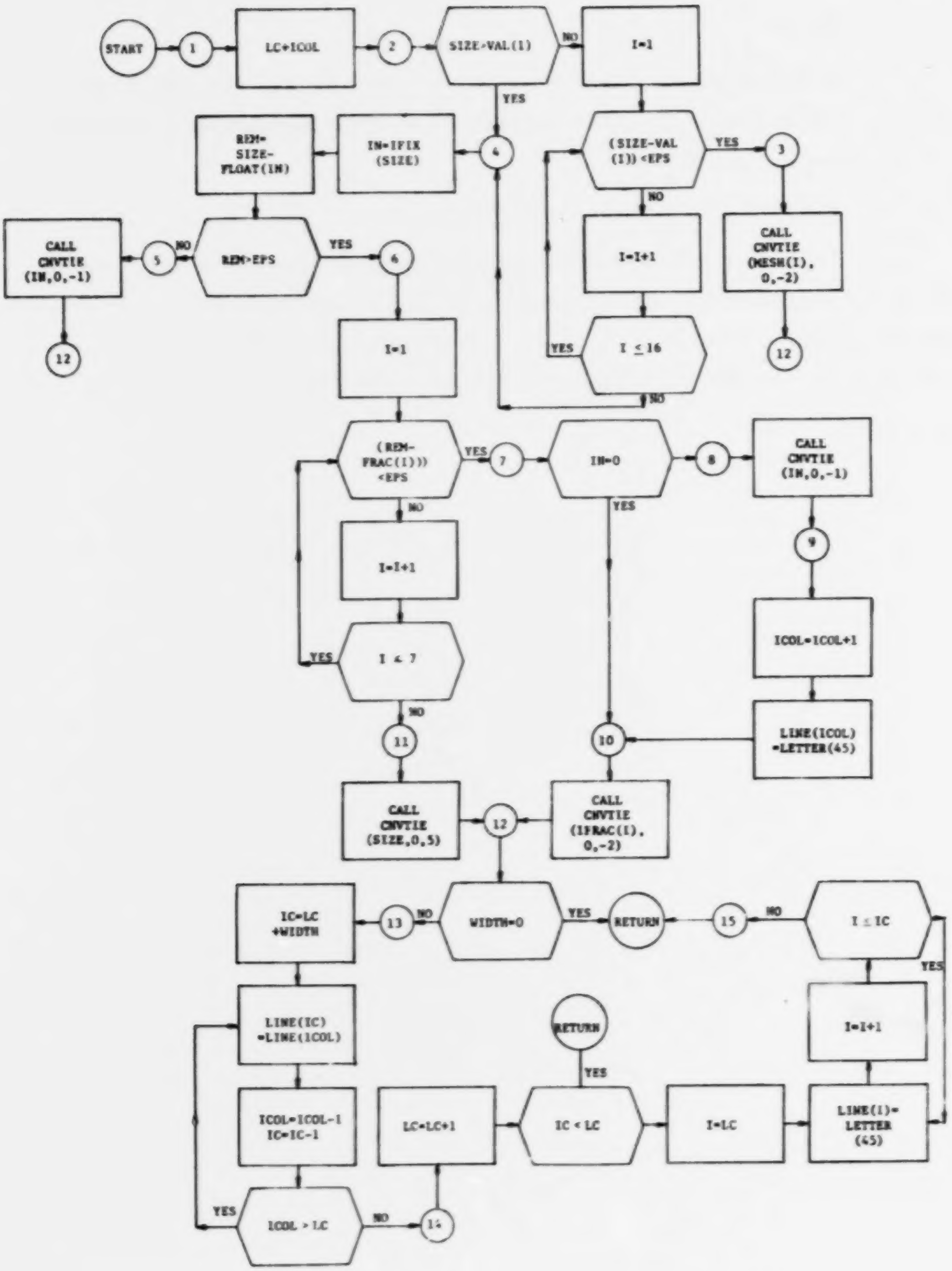

START
1
LC=ICOL
2
SIZE>VAL(1)
NO
I=1
YES
4
IN=IFIX (SIZE)
REM= SIZE- FLOAT(IN)
(SIZE-VAL (I)) <EPS
YES
3
NO
CALL CNVTIE (MESH(I), 0,-2)
12
I=I+1
I ≤ 16
YES
NO
CALL CNVTIE (IN,0,-1)
5
NO
REM>EPS
YES
6
12
I=1
(REM- FRAC(I))) <EPS
YES
7
IN=0
8
CALL CNVTIE (IN,0,-1)
NO
YES
9
I=I+1
ICOL=ICOL+1
I ≤ 7
YES
NO
11
10
LINE(ICOL) =LETTER(45)
CALL CNVTIE (SIZE,0,5)
12
CALL CNVTIE (IFRAC(I), 0,-2)
IC=LC +WIDTH
13
NO
WIDTH=0
YES
RETURN
15
NO
I ≤ IC
YES
LINE(IC) =LINE(ICOL)
RETURN
YES
I=I+1
ICOL=ICOL-1 IC=IC-1
LC=LC+1
IC < LC
I=LC
LINE(I)= LETTER (45)
YES
ICOL > LC
NO
14

FIGURE II-6-9.3. FLOWCHART FOR SUBROUTINE PRSIZE

Reproduced from best available copy.

```
      SUBROUTINE PRSIZE(SIZE,WIDTH)
C
C     TITLE(PRINT SIZE VALUE)
C
C     SUBROUTINE PRSIZE IS USED TO CONTROL THE PRINTING OF SIZE VALUES.
C     THIS ROUTINE ASSUMES THAT ALL SIZE VALUES ARE INTERNALLY REPRE-
C     SENTED AS FLOATING POINT INCHES. IF THAT FLOATING POINT VALUE
C     CORRESPONDS TO ONE OF THE STANDARD MESH SIZES, THEN IT IS DIS-
C     PLAYED AS SUCH FOLLOWED BY THE UNITS INDICATOR (M). IF THE SIZE
C     VALUE IS AN EVEN FRACTION IN HALVES, FORTHS, OR IN EIGHTS, THEN
C     THE ROUTINE PRINTS IT IN FRACTIONAL FORM. IN ALL OTHER CASES THE
C     ROUTINE PRINTS THE VALUE IN DECIMAL FORM WITH UP TO FIVE DECIMAL
C     PLACES. THE ROUTINE WILL EITHER RIGHT OR LEFT JUSTIFY THE DISPLAY,
C     DEPENDING UPON THE CONTEXT.
C
C     PARAMETER    DESCRIPTION
C     SIZE         THIS REAL VARIABLE CONTAINS THE SIZE VALUE TO BE
C                  DISPLAYED.
C     WIDTH        THIS INTEGER VARIABLE CONTAINS THE WIDTH OF THE FIELD
C                  TO CONTAIN THE DISPLAY. IF IT IS GREATER THAN ZERO,
C                  THEN THE DISPLAY IS RIGHT JUSTIFIED IN A FIELD OF
C                  THE SPECIFIED WIDTH. IF IT IS ZERO, THEN THE DIS-
C                  PLAY IS LEFT-JUSTIFIED IN A FIELD OF WHATEVER WIDTH
C                  IS NEEDED TO CONTAIN THE DISPLAY CHARACTERS GENERATED.
C
C     COMMON       VARIABLES USED
C     /CODERC/     ICOL,LINE
C     /CHRSET/     LETTER
C
C     EXTERNALS    ABS, CNVTIE, IFIX, FLOAT
C
C     VARIABLE     DESCRIPTION
C     VAL          THIS REAL 1-DIMENSIONAL ARRAY CONTAINS THE SIZES IN
C                  INCHES OF THE STANDARD MESH SIZES.
C     MESH         THIS INTEGER 1-DIMENSIONAL ARRAY CONTAINS THE MESH
C                  CODES STORED AS CHARACTER SEQUENCES BASE 47.
C     FRAC         THIS REAL 1-DIMENSIONAL ARRAY CONTAINS THE VALUES
C                  FOR THE EVEN FRACTIONS RECOGNIZED BY THE ROUTINE.
C     IFRAC        THIS INTEGER 1-DIMENSIONAL ARRAY CONTAINS THE DIS-
C                  PLAY CODES NEEDED FOR THE FRACTIONS STORED AS NUM-
C                  BERS BASE 47.
C     EPS          THIS REAL VARIABLE CONTAINS THE THRESHHOLD VALUE
C                  FOR REAL EQUALITY COMPARISONS.
C     LC           THIS INTEGER VARIABLE CONTAINS THE STARTING COLUMN
C                  POSITION IN THE DISPLAY WHEN THIS ROUTINE WAS CALLED.
C     I            THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                  MOVEMENT THROUGH LOCAL LOOPS.
C     IN           THIS INTEGER VARIABLE CONTAINS THE NONDECIMAL VALUE
C                  OF THE SIZE.
```

FIGURE II-6-9.4. FORTRAN LISTING FOR SUBROUTINE PRSIZE

```
C     REM            THIS REAL VARIABLE CONTAINS THE DECIMAL PORTION OF
C                    THE SIZE.
C     IC             THIS INTEGER VARIABLE IS USED TO CONTAIN THE ENDING
C                    POSITION OF THE DISPLAY.
C
      DIMENSION VAL(16),MESH(16),FRAC(7),IFRAC(7)
      COMMON/CODERC/ICOL,LCOL,LINE(150)
      COMMON/CHRSET/LETTER(47)
      INTEGER WIDTH
      DATA EPS/0.000001/
      DATA MESH(1),MESH(2),MESH(3),MESH(4)/1997,2125,2253,116429/
      DATA MESH(5),MESH(6),MESH(7),MESH(8)/116685,120525,121037,124941/
      DATA MESH(9),MESH(10),MESH(11)         /129229,137229,7452365/
      DATA MESH(12)/7472845/
      DATA MESH(13),MESH(14),MESH(15)/7714509,7743181,7985165/
      DATA MESH(16)/8238797/
      DATA IFRAC(1),IFRAC(2),IFRAC(3)/141859,125471,133667/
      DATA IFRAC(4),IFRAC(5),IFRAC(6)/117277,125475,117279/
      DATA IFRAC(7)/117283/
      DATA VAL(1),VAL(2),VAL(3),VAL(4)/0.185,0.131,0.093,0.065/
      DATA VAL(5),VAL(6),VAL(7),VAL(8)/0.046,0.0328,0.0232,0.0164/
      DATA VAL(9),VAL(10),VAL(11),VAL(12)/0.0116,0.0082,0.0058,0.0041/
      DATA VAL(13),VAL(14),VAL(15),VAL(16)/0.0029,0.0021,0.0017,0.0015/
      DATA FRAC(1),FRAC(2),FRAC(3),FRAC(4)/0.875,0.75,0.625,0.5/
      DATA FRAC(5),FRAC(6),FRAC(7)/0.375,0.25,0.125/
C
C     STEP 1         TRAP THE CURRENT VALUE OF THE CHARACTER POSITION
C                    COUNTER.
C
      LC=ICOL
C
C     STEP 2         IF THE SIZE VALUE CORRESPONDS TO ONE OF THE MESH
C                    SIZES BRANCH TO STEP 3. ELSE, BRANCH TO STEP 4.
C
      IF(SIZE.GT.VAL(1))GO TO 20
      DO 10 I=1,16
      IF(ABS(SIZE-VAL(I)).LT.EPS)GO TO 15
   10 CONTINUE
      GO TO 20
C
C     STEP 3         THE SIZE IS A STANDARD MESH SIZE. DISPLAY THE STAN-
C                    DARD MESH CODE CORRESPONDING TO THE VALUE AND BRANCH
C                    TO STEP 12.
C
   15 CALL CNVTIF(MESH(I),0,-2)
      GO TO 55
C
C     STEP 4         OBTAIN THE DECIMAL AND NONDECIMAL PORTIONS OF THE
C                    SIZE VALUE. IF THE DECIMAL PORTION IS NONZERO,
```

FIGURE II-6-9.4. (Continued)

```
C                    BRANCH TO STEP 6.
C
   20 IN=IFIX(SIZE)
      REM=SIZE-FLOAT(IN)
      IF(REM.GT.EPS)GO TO 25
C
C     STEP 5         THE SIZE REPRESENTS AN EVEN NUMBER OF INCHES. DIS-
C                    PLAY THEM AND BRANCH TO STEP 12.
C
      CALL CNVTIE(IN,0,-1)
      GO TO 55
C
C     STEP 6         DETERMINE IF THE DECIMAL PORTION REPESENTS AN EVEN
C                    FRACTION. IF NOT, BRANCH TO STEP 11.
C
   25 DO 30 I=1,7
      IF(ABS(REM-FRAC(I)).LT.EPS)GO TO 35
   30 CONTINUE
      GO TO 45
C
C     STEP 7         THE DECIMAL PORTION IS AN EVEN FRACTION. IF THE
C                    NONDECIMAL PORTION IS ZERO, BRANCH TO STEP10.
C
   35 IF(IN.EQ.0)GO TO 40
C
C     STEP 8         DISPLAY THE NONDECIMAL PORTION OF THE FRACTION USING
C                    SUBROUTINE  C N V D I S  .
C
      CALL CNVTIE(IN,0,-1)
C
C     STEP 9         INSERT A BLANK BETWEEN THE NONFRACTIONAL PART AND
C                    THE FOLLOWING FRACTIONAL PART.
C
      ICOL=ICOL+1
      LINE(ICOL)=LETTER(45)
C
C     STEP 10        USING SUBROUTINE  C N V D I S  DISPLAY THE FRACTIONAL
C                    PART OF THE SIZE VALUE AND BRANCH TO STEP 12.
C
   40 CALL CNVTIE(IFRAC(I),0,-2)
      GO TO 55
C
C     STEP 11        THE SIZE VALUE DOES NOT REPRESENT ANY EVEN VALUE.
C                    USING SUBROUTINE  C N V V A L  DISPLAY THE VALUE
C                    WITH UP TO 5 DECIMAL PLACES.
C
   45 CALL CNVTIE(SIZE,0,5)
C
C     STEP 12        THE DISPLAY HAS BEEN LEFT-JUSTIFIED. IF THAT IS THE
```

FIGURE II-6-9.4. (Continued)

```
C                   DESIRED ORIENTATION, END LOCAL PROCESSING.
C
   55 IF(WIDTH.EQ.0)GO TO 999
C
C     STEP 13       THE CHARACTERS MUST BE RIGHT JUSTIFIED. MOVE THEM
C                   TO THE RIGHT THE APPROPRIATE NUMBER OF PLACES.
C
      IC=LC+WIDTH
   60 LINE(IC)=LINE(ICOL)
      ICOL=ICOL-1
      IC=IC-1
      IF(ICOL.GT.LC)GO TO 60
      ICOL=LC+WIDTH
C
C     STEP 14       IF THE LEFT-HAND SIDE OF THE FIELD CONTAINS SPACES,
C                   BLANK THESE OUT.
C
      LC=LC+1
      IF(IC.LT.LC)GO TO 999
      DO 65 I=LC,IC
      LINE(I)=LETTER(45)
   65 CONTINUE
C
C     STEP 15       END LOCAL PROCESSING.
C
  999 RETURN
      END
```

FIGURE II-6-9.4. (Continued)

6.10 Subroutine RDDESC

Another problem in CPSM4 requiring some character manipulation capability involved the ability to display the unit descriptions in various types of contexts. This problem was introduced in part 6.2 of this chapter. Subroutine RDDESC is used to read the descriptor into a linear array. Given the previous discussion, the methodology used is straightforward from the basic documentation which is shown in Figure II-6-10.1. The routine does little non-local looping; therefore, no pictorial flowchart is included.

```
      SUBROUTINE RDDESC(DESC,NCELL)
C
C     TITLE(READ DESCRIPTOR)
C
C     SUBROUTINE RDDESC IS USED TO READ A DESCRIPTOR CONTAINED WITHIN
C     A SPECIFIED FIELD AND STORED IN LITERAL FORM AND TO CONVERT IT
C     INTO A N-PRECISION NUMBER REPRESENTED BASE 64.
C
C     PARAMETER     DESCRIPTION
C     DESC          THIS 1-DIMENSIONAL, INTEGER ARRAY RETURNS THE BASE
C                   64 NUMBER REPRESENTING THE DESCRIPTOR. IT IS STORED
C                   AS FOLLOWS.
C                   CELL      DESCRIPTION
C                    1   THE NUMBER OF DIGITS IN THE DESCRIPTOR.
C                    2+  THE ACTUAL NUMBER STORED 5 DIGITS PER CELL.
C     NCELL         THIS INTEGER VARIABLE RETURNS THE NUMBER OF CELLS
C                   USED IN THE DESCRIPTOR ARRAY.
C
C     COMMON        VARIABLES USED
C     /CODERC/      ICOL, LCOL, LINE
C     /CHRSET/      LETTER
C
C     VARIABLE      DESCRIPTION
C     I             THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                   MOVEMENT THROUGH THE CHARACTER ARRAY.
C     J             THIS INTEGER VARIABLE IS USED TO CONTAIN THE SEQUENCE
C                   NUMBER OF THE CURRENT CHARACTER BEING EXAMINED.
C     LC            THIS INTEGER VARIABLE CONTAINS THE RELATIVE ADDRESS
C                   IN THE CHARACTER ARRAY OF THE LAST NONBLANK CHAR-
C                   ACTER.
C
      COMMON/CODERC/ICOL,LCOL,LINE(150)
      COMMON/CHRSET/LETTER(47)
      DIMENSION DESC(1)
      INTEGER DESC
C
C     STEP 1        CONVERT THE CHARACTERS IN THE RAW INPUT FIELD INTO
C                   THEIR INTERNAL DISPLAY CODE REPRESENTATIONS.
C
      DO 30 I=ICOL,LCOL
      IF(LINE(I)-LETTER(45))10,20,10
   10 DO 15 J=1,47
      IF(LINE(I)-LETTER(J))15,25,15
   15 CONTINUE
   20 LINE(I)=45
      GO TO 30
   25 LINE(I)=J
   30 CONTINUE
C
```

FIGURE II-6-10.1. FORTRAN LISTING FOR SUBROUTINE RDDESC

```
C     STEP 2        INITIALIZE THE NUMBER DIGITS IN THE DESCRIPTOR AT
C                   ZERO, AND THE NUMBER OF CELLS USED AT 1.
C
      DESC(1)=0
      NCELL=1
C
C     STEP 3        FIND THE TRAILING NON-BLANK CHARACTER IN THE INPUT
C                   FIELD. IF THE FIELD IS ALL BLANK, END LOCAL PRO-
C                   CESSING.
C
      LC=LCOL
   35 IF(LINE(LC)-45)50,40,50
   40 LC=LC-1
      IF(LC)999,999,35
C
C     STEP 4        ENTER THE NUMBER OF CHARACTERS IN THE DESCRIPTOR IN
C                   WORD 1 OF THE DESCRIPTOR RETURN ARRAY.
C
   50 CONTINUE
      DESC(1)=LC-ICOL+1
C
C     STEP 5        THE NEXT BLOCK OF DIGITS MAY NOW BE STORED. INCRE-
C                   MENT THE NUMBER OF CELLS USED BY ONE AND INITIALIZE
C                   THE NEW CELL AT ZERO.
C
   55 NCELL=NCELL+1
      DESC(NCELL)=0
C
C     STEP 6        INITIALIZE THE DUMMY COUNTER TO CONTROL THE PACKING
C                   OF THE CHARACTERS AT ZERO.
C
      DO 60 I=1,5
C
C     STEP 7        STORE THE NEXT DIGIT IN THE CURRENT CELL.
C
      DESC(NCELL)=DESC(NCELL)*64+LINE(ICOL)
C
C     STEP 8        INCREMENT THE ADDRESS OF THE NEXT CHARACTER BY ONE,
C                   AND IF ALL CHARACTERS HAVE BEEN PROCESSED, END
C                   LOCAL PROCESSING.
C
      ICOL=ICOL+1
      IF(LC-ICOL)999,60,60
C
C     STEP 9        INCREMENT THE DUMMY COUNTER CONTROLING THE PACKING
C                   OF THE CURRENT CELL BY 1, AND IF IT DOES NOT EXCEED
C                   5 LOOP BACK TO STEP 7. IF IT DOES, LOOP BACK TO
C                   STEP 5.
C
```

FIGURE II-6-10.1. (Continued)

```
   60 CONTINUE
      GO TO 55
  999 RETURN
      END
```

FIGURE II-6-11.1. (Continued

6.11 Subroutine PRDESC

The routine that reconverts stored descriptors back into display form is subroutine PRDESC. The routine is straightforward, given the previous discussion. Figure II-6-11.1 contains the listing and basic documentation. The logical flow of the routine is very simple; therefore, no pictorial flowchart is included.

```
      SUBROUTINE PRDESC(DESC)
C
C     TITLE(PRINT DESCRIPTOR)
C
C     SUBROUTINE PRDESC IS USED TO CONTROL THE PRINTING OF DESCRIPTORS.
C     IN OTHER WORDS, THIS ROUTINE CONVERTS A DESCRIPTOR STORED IN STAN-
C     DARD STRING FORM INTO A SEQUENCE OF CHARACTERS WHICH MAY BE
C     PRINTED.
C
C     PARAMETER     DESCRIPTION
C     DESC          THIS 1-DIMENSIONAL INTEGER ARRAY CONTAINS THE DES-
C                   CRIPTOR TO BE PRINTED STORED IN STANDARD STRING
C                   FORMAT.
C
C     COMMON        VARIABLES USED
C     /CHRSET/      LETTER
C     /CODERC/      ICOL, LINE
C
C     VARIABLE      DESCRIPTION
C     IW            THIS DUMMY INTEGER VARIABLE IS USED TO CONTAIN THE
C                   RELATIVE ADDRESS IN THE STRING ARRAY OR THE CURRENT
C                   WORD BEING PROCESSED.
C     NC            THIS INTEGER VARIABLE CONTAINS THE NUMBER OF CHAR-
C                   ACTERS YET TO BE REMOVED FROM THE STRING.
C     NPW           THIS INTEGER VARIABLE CONTAINS THE NUMBER OF CHAR-
C                   ACTERS WITHIN THE CURRENT WORD OF THE STRING.
C     IWORD         THIS INTEGER VARIABLE CONTAINS THE CURRENT WORD OF
C                   THE STRING BEING PROCESSED.
C     IC            THIS TEMPORARY INTEGER VARIABLE CONTAINS THE PAR-
C                   TIALLY UNPACKED CHARACTERS OF THE CURRENT WORD OF
C                   THE STRING BEING PROCESSED.
C     LET           THIS INTEGER VARIABLE CONTAINS THE SEQUENCE NUMBER OF
C                   THE CURRENT CHARACTER UNPACKED FROM THE STRING.
C     I             THIS DUMMY INTEGER VARIABLE IS USED AS A DO-LOOP
C                   COUNTER.
C
      DIMENSION DESC(1)
      COMMON/CHRSET/LETTER(47)
      COMMON/CODERC/ICOL,LCOL,LINE(150)
      INTEGER DESC
C
C     STEP 1        INITIALIZE THE SUBSCRIPT FOR THE STRING ARRAY,
C                   OBTAIN THE NUMBER OF CHARACTERS IN THE STRING, AND
C                   INITIALIZE THE NUMBER OF CHARACTERS PACKED INTO THE
C                   FIRST WORD AT THE MAXIMUM NUMBER OF CHARATERS ALLOWED
C                   PER WORD.
C
      IW=1
      NC=DESC(IW)
```

FIGURE II-6-11.1. FORTRAN LISTING FOR SUBROUTINE PRDESC

Reproduced from
best available copy.

```
      NPW=5
C
C     STEP 2        IF THE NUMBER OF CHARACTERS REMAINING IN THE STRING
C                   IS LESS THAN THE NUMBER OF CHARACTERS IN THE CURRENT
C                   WORD, SET THE NUMBER OF CHARACTERS IN THE WORD EQUAL
C                   TO THE NUMBER OF CHARACTERS REMAINING IN THE STRING.
C
   10 IF(NPW.GT.NC)NPW=NC
C
C     STEP 3        UNPACK AND STORE IN THE OUTPUT LINE THE CHARACTERS
C                   STORED IN THE NEXT WORD OF THE STRING.
C
      IW=IW+1
      ICOL=ICOL+NPW
      IWORD=DESC(IW)
      DO 15 I=1,NPW
      IC=IWORD/64
      LET=IWORD-IC*64
      LINE(ICOL)=LETTER(LET)
      ICOL=ICOL-1
      IWORD=IC
   15 CONTINUE
C
C     STEP 4        INCREMENT THE ADDRESS IN THE OUTPUT ARRAY OF THE
C                   LAST CHARACTER STORED BY THE NUMBER OF CHARACTERS
C                   IN THE LAST PROCESSED WORD, AND DECREMENT THE
C                   NUMBER OF CHARACTERS REMAINING IN THE STRING BY THAT
C                   SAME AMOUNT.
C
      ICOL=ICOL+NPW
      NC=NC-NPW
C
C     STEP 5        IF THERE ARE ANY CHARACTERS REMAINING IN THE STRING,
C                   LOOP BACK TO STEP 2.
C
      IF(NC.GT.0)GO TO 10
C
C     STEP 6        END LOCAL PROCESSING.
C
      RETURN
      END
```

FIGURE II-6-11.1. (Continued)

6.12 Subroutine RDVAL

Subroutine RDVAL converts a coded character string representing an integer into the numeric value of the integer. The listing and basic documentation for this routine are given in Figure II-6-12.1.

```
      SUBROUTINE RDVAL(IVAL,IBLANK)
C
C     TITLE(RDVAL CONVERTS A CODED INTEGER VALUE INTO ITS NUMERIC VALUE)
C
C     SUBROUTINE RDVAL COMPUTES THE NUMERIC VALUE OF A CODED CHARACTER
C     STRING REPRESENTING AN INTEGER.
C
C     PARAMETER      DESCRIPTION
C     IVAL           THIS INTEGER VARIABLE RETURNS THE INTEGER VALUE OF
C                    THE STRING.
C     IBLANK         THIS INTEGER VARIABLE CONTAINS A VALUE TO BE USED
C                    TO INDICATE A BLANK STRING.
C
C     COMMON         VARIABLES USED
C     /IOCNT/        CARCON,JOB
C     /CHRSET/       LETTER
C     /CODERC/       ICOL,LCOL,LINE
C
C     EXTERNALS      PRDESC, PRLINE
C
C     VARIABLE       DESCRIPTION
C     I,J            THESE INTEGER VARIABLES ARE USED AS DO-LOOP COUNTERS.
C     IC             THIS INTEGER VARIABLE IS USED TO INDICATE THE
C                    STARTING POSITION OF THE DIGITS TO BE DECODED.
C     LD1            THIS INTEGER ARRAY CONTAINS PACKED CHARACTER CODES
C                    FOR DISPLAY.
C     MINUS          THIS INTEGER VARIABLE IS USED TO INDICATE A NEGATIVE
C                    NUMBER
C
      COMMON/CHRSET/LETTER(47)
      COMMON/FILES/READER,PRINTR,OFFLIN
      COMMON/CODERC/ICOL,LCOL,LINE(150)
      COMMON/IOCNT/JOB,CARCON
      INTEGER CARCON
      INTEGER READER,PRINTR,OFFLIN
      DIMENSION LD1(8)
      DATA LD1(1),LD1(2),LD1(3),LD1(4)/34,154190151,20107464,21500116/
      DATA LD1(5),LD1(6),LD1(7)/88789582,758731589,304364358/
      DATA LD1(8)/2360549/
      MINUS=1
C     STEP 1         DETERMINE THE FIRST NON-BLANK CHARACTER.  IF ALL
C                    CHARACTERS ARE BLANK, BRANCH TO STEP 7.
      DO 10 I=ICOL,LCOL
      IF(LINE(I).NE.LETTER(45))GO TO 15
   10 CONTINUE
      IVAL=IBLANK
      GO TO 999
C     STEP 2         DETERMINE WHETHER THE NUMBER IS NEGATIVE.
   15 IVAL=0
```

FIGURE II-6-12.1. LISTING FOR SUBROUTINE RDVAL

```
      IF(LINE(I).NE.LETTER(38))GO TO 16
      MINUS=-1
      I=I+1
C     STEP 3        DETERMINE THE NUMERIC VALUE OF THE STRING.
   16 IC=I
      DO 30 I=IC,LCOL
      DO 20 J=27,36
      IF(LINE(I).EQ.LETTER(J))GO TO 25
   20 CONTINUE
      GO TO 35
   25 IVAL=IVAL*10+J-27
   30 CONTINUE
C     STEP 4        BRANCH TO STEP 7.
      GO TO 999
C     STEP 5        CHECK WHETHER THE REMAINING CHARACTERS ARE BLANK.
C                   IF SO, BRANCH TO STEP 7.
   35 IC=I
      DO 40 I=IC,LCOL
      IF(LINE(I).NE.LETTER(45))GO TO 998
   40 CONTINUE
      GO TO 999
C     STEP 6        NON-NUMERIC CHARACTERS HAVE BEEN FOUND.  PRINT AN
C                   ERROR MESSAGE AND END PROCESSING.
  998 CARCON=LETTER(28)
      JOB=-1
      ICOL=0
      CALL PRDESC(LD1)
      CALL PRLINE
      STOP
C     STEP 7        IF THE NUMBER IS NEGATIVE, TAKE THE NEGATIVE VALUE.
  999 IF(MINUS.LT.0)IVAL=-IVAL
C     STEP 8        END LOCAL PROCESSING.
      RETURN
      END
```

FIGURE II-6-12.1. (Continued)

7.0 INPUT PROCESSING

The purpose of the input processing routines is to read the user input from either data cards or from data bases, to create data bases of the feed and configuration data if these do not already exist, and to store the data in the program arrays. The presentation in this chapter relies heavily on the material presented in Chapter 6 of Volume I which describes the preparation of the input and the construction of the data bases, and on the material in Chapter 2 of Volume II which describes how information is stored in the program.

7.1 Subroutine RDUNIT

Subroutine RDUNIT reads the unit description cards. The listing and basic documentation for this routine are given in Figure II-7-1.1. Note that RDUNIT makes certain that the equipment identifier associated with the unit is in fact defined within the equipment inventory. If this is not true, the routine ends the execution of the program after issuing an error message.

Reproduced from best available copy.

```
      SUBROUTINE RDUNIT
C
C     TITLE(READ UNIT INFORMATION)
C
C     SUBROUTINE RDUNIT IS USED TO CONTROL THE READING AND STORING OF
C     THE UNIT INFORMATION FOR THE CONFIGURATION.
C
C     EXTERNALS     RDDESC
C
C     COMMON        VARIABLES USED
C     /WORK/        NEXT, IUNIT, ICURVE
C     /FILES/       READER, PRINTR
C     /CODERC/      ICOL, LCOL, LINE
C     //            WS
C     /CNTVAR/      L, D1, D2, D3
C
C     VARIABLE      DESCRIPTION
C     IU            THIS INTEGER VARIABLE CONTAINS THE ADDRESS IN WORK-
C                   ING STORAGE INTO WHICH THE UNIT DESCRIPTOR ADDRESS
C                   IS TO BE PLACED.
C     U, I          THESE DUMMY INTEGER VARIABLES ARE USED TO CONTROL THE
C                   MOVEMENT THROUGH THE UNITS IN THE CONFIGURATION.
C     IEQUIP        THIS INTEGER VARIABLE CONTAINS THE ADDRESS IN WORK-
C                   STORAGE OF THE EQUIPMENT INFORMATION BLOCK ASSOCI-
C                   ATED WITH THE UNIT.
C     ID            THIS INTEGER VARIABLE CONTAINS THE EQUIPMENT IDEN-
C                   TIFIER ASSOCIATED WITH THE UNIT.
C     IUO           THIS INTEGER VARIABLE CONTAINS THE UNIT OPERATION
C                   TYPE CODE ASSOCIATED WITH THE UNIT.
C     NCELL         THIS INTEGER VARIABLE CONTAINS THE NUMBER OF CELLS
C                   OF WORKING STORAGE REQUIRED BY THE UNIT DESCRIPTOR.
C
      COMMON/WORK/NEXT,LAST,ICURVE,IUNIT,IDATA,ITITLE,ICHAR,IFLW
      COMMON/FILES/READER,PRINTR,OFFLIN
      COMMON/CODERC/ICOL,LCOL,LINE(40),CA(110)
      COMMON WS(6000)
      COMMON/CNTVAR/KF(2,100),SIZE(23),SBOUND(21),GRAV(20)
     1,WFLOW(100) ,S(100),D1(60),D2(60),D3(60),L(60)
     2                          ,NSIZE,NGRAV,IOUT,NUNITS,NFLOWS,NCHAR
      INTEGER D1,D2,D3
      INTEGER WS,READER,PRINTR,U
    2 FORMAT(2I5,3F10.0,40A1)
C
C     STEP 1        ALLOCATE SPACE FOR, INITIALIZE, AND RECORD THE
C                   ADDRESS IN WORKING STORAGE OF THE UNIT INFORMATION
C                   CONTROL VECTOR.
C
      IUNIT=NEXT
      NEXT=NEXT+NUNITS
```

FIGURE II-7-1.1. LISTING FOR SUBROUTINE RDUNIT

```
      IU=IUNIT-1
C
C     STEP 2         INITIALIZE AT ONE THE DUMMY COUNTER TO CONTROL THE
C                    MOVEMENT THROUGH THE UNITS.
C
      DO 80 U=1,NUNITS
C
C     STEP 3         UPDATE THE ADDRESS OF THE CURRENT UNIT IFORMATION
C                    ADDRESS CELL.
C
      IU=IU+1
C
C     STEP 4         READ IN THE BASIC INFORMATION FOR THE CURRENT UNIT.
C
      READ(READER,2) L(U),D1(U),O1(U),D2(U),D3(U),LINE
C
C     STEP 5         LOCATE THE EQUIPMENT INFORMATION BLOCK IN WORKING
C                    STORAGE FOR THAT EQUIPMENT TYPE IDENTIFIED BY THE
C                    UNIT. IF NO SUCH BLOCK CAN BE FOUND, END THE RUN
C                    WITH AN APPROPRIATE MESSAGE.
C
      ID=L(U)
      IEQUIP=ICURVE
   20 IF(ID.EQ.WS(IEQUIP-1))GO TO 25
      IEQUIP=WS(IEQUIP)
      IF(IEQUIP.GT.0)GO TO 20
      WRITE(PRINTR,6)ID
    6 FORMAT(1H1,40HTHERE IS NO EQUIPMENT TYPE IDENTIFIED BY,I4)
      STOP
C
C     STEP 6         RECORD THE BASE ADDRESS OF THE EQUIPMENT TYPE OF
C                    THE UNIT.
C
   25 L(U)=IEQUIP
C
C     STEP 7         OBTAIN THE UNIT OPERATION TYPE CODE FOR THE EQUIP-
C                    MENT TABLE.
C
      IUO=WS(IEQUIP+2)
C
C     STEP 8         READ AND STORE THE UNIT DESCRIPTOR. IF NO DESCRIPTOR
C                    IS PROVIDED FOR THE UNIT, ASSIGN IT THE EQUIPMENT
C                    DESCRIPTOR.
C
      ICOL=1
      LCOL=40
      CALL RDDESC(WS(NEXT),NCELL)
      WS(IU)=WS(IEQUIP+1)
      IF(WS(NEXT).EQ.0) GOTO 80
```

FIGURE II-7-1.1. (Continued)

```
      WS(IU)=NEXT
      NEXT=NEXT+NCELL
C
C
C     STEP 9          INCREMENT BY ONE THE DUMMY COUNTER CONTROLLING THE
C                     MOVEMENT THROUGH THE UNITS, AND IF ALL HAVE NOT
C                     BEEN PROCESSED, LOOP BACK TO STEP 3.
C
   80 CONTINUE
C
C     STEP 10         STORE THE DECISION VARIABLES AND IDENTIFIERS FOR
C                     EACH UNIT.
C
      IUNIT=NEXT-IUNIT
      DO 90 I=1,NUNITS
      WS(NEXT)=L(I)
      NEXT=NEXT+1
      WS(NEXT)=D1(I)
      NEXT=NEXT+1
      WS(NEXT)=D2(I)
      NEXT=NEXT+1
      WS(NEXT)=D3(I)
      NEXT=NEXT+1
   90 CONTINUE
C
C     STEP 11         END LOCAL PROCESSING.
C
      RETURN
      END
```

FIGURE II-7-1.1. (Continued)

7.2 Subroutine RDCURV

Of all the input routines, subroutine RDCURV is by far the most complex. Its role is to read the equipment inventory and to store it in working storage. The listing and basic documentation for this routine is given in Figure II-7-2.1. The material presented there depends heavily on the discussion in Volume I, Chapter 5, Part 3 and on the information presented in Figure II-2-2.1.

```
      SUBROUTINE RDCURV
C
C     TITLE(READ EQUIPMENT CURVE DATA)
C     SUBROUTINE RDCURV IS USED TO CONTROL THE READING OF THE EQUIP-
C     MENT CURVE DATA AND ASSOCIATED CONSTANTS. THE SUM TOTAL OF THIS
C     INFORMATION DEFINES THE EQUIPMENT INVENTORY AVAILABLE TO THE
C     SIMULATION.
C
C     EXTERNALS    RDDESC
C
C     COMMON       VARIABLES USED
C     /FILES/      READER
C     /WORK/       NEXT, ICURVE
C     /CODERC/     ICOL, LCOL, LINE
C     //           WS
C
C     VARIABLE     DESCRIPTION
C     NCONS        THIS INTEGER, 1-DIMENSIONAL ARRAY CONTAINS THE NUMBER
C                  OF CONSTANTS TO BE READ IN FOR EACH UNIT OPERATION
C                  TYPE.
C     CONS, ICONS  THIS TYPELESS, 1-DIMENSIONAL ARRAY IS USED AS A
C                  SCRATCH STORAGE ARRAY INTO WHICH THE UNIT OPERATION
C                  CONSTANTS ARE FIRST READ.
C     NCURVE       THIS INTEGER, 1-DIMENSIONAL ARRAY CONTAINS THE NUM-
C                  BER OF CURVE SETS TO BE READ IN FOR EACH UNIT
C                  OPERATION TYPE.
C     NEQUIP       THIS INTEGER VARIABLE CONTAINS THE NUMBER OF EQUIP-
C                  MENT TYPES IN THE ENTIRE INVENTORY.
C     IE           THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                  MOVEMENT THROUGH THE INVENTORY.
C     IEQUIP       THIS INTEGER VARIABLE CONTAINS THE ADDRESS IN WORK-
C                  STORAGE OF THE CURRENT EQUIPMENT INFORMATION BLOCK.
C     ID           THIS INTEGER VARIABLE CONTAINS THE EQUIPMENT INVEN-
C                  TORY IDENTIFIER.
C     IUO          THIS INTEGER VARIABLE CONTAINS THE UNIT OPERATION
C                  TYPE CODE ASSOCIATED WITH THE EQUIPMENT TYPE.
C     IDESC        THIS INTEGER VARIABLE CONTAINS THE ADDRESS IN WORKING
C                  STORAGE OF THE EQUIPMENT DESCRIPTOR.
C     NCELL        THIS INTEGER VARIABLE CONTAINS THE NUMBER OF CELLS
C                  OF WORKING STORAGE REQUIRED TO STORE THE EQUIPMENT
C                  DESCRIPTOR.
C     NC           THIS INTEGER VARIABLE CONTAINS THE NUMBER OF CON-
C                  STANTS AND THEN CURVE SETS TO BE READ IN FOR THE
C                  CURRENT EQUIPMENT TYPE.
C     I            THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                  MOVEMENT THROUGH LOCAL LOOPS.
C     ICV          THIS INTEGER VARIABLE CONTAINS THE ADDRESS IN
C                  WORKING STORAGE OF THE CURRENT CURVE SET INFORMATION
C                  BLOCK.
```

FIGURE II-7-2.1. LISTING FOR SUBROUTINE RDCURV

```
C     ITYPE          THIS INTEGER VARIABLE CONTAINS THE TYPE CODE FOR THE
C                    CURRENT CURVE SET.
C     NPOINT         THIS INTEGER VARIABLE CONTAINS THE NUMBER OF POINTS
C                    PER CURVE IN THE CURVE SET.
C     NSIZE          THIS INTEGER VARIABLE CONTAINS THE NUMBER OF CURVES
C                    OR VECTORS OR SIZE FRACTIONS IN THE CURRENT SET.
C     LCELL          THIS DUMMY INTEGER VARIABLE IS USED TO SPECIFIY THE
C                    ENDING ADDRESS IN WORKING STORAGE TO WHICH A READ
C                    SHOULD BE PERFORMED.
C     NCRV           THIS INTEGER VARIABLE CONTAINS THE ACTUAL NUMBER OF
C                    CURVES OR VECTORS WITHIN THE CURVE SET.
C     IX             THIS INTEGER VARIABLE CONTAINS THE ADDRESS IN WORKING
C                    STORAGE OF THE VECTOR OF INDEPENDENT VALUES TO BE
C                    READ IN.
C     IY             THIS INTEGER VARIABLE CONTAINS THE ADDRESS IN WORK-
C                    ING STORAGE OF THE DEPENDENT VALUES TO BE READ IN.
C     IC             THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                    MOVEMENT THROUGH THE CURVE SETS.
C     ICC            THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                    MOVEMENT THROUGH THE ACTUAL CURVES WITHIN THE CUR-
C                    RENT CURVE SET.
C     IBR            THIS INTEGER VARIANLE DETERMINES WHETHER THE FLOW IS
C                    SUBJECT TO BREAKAGE DURING OPERATION OF THE UNIT.
C     NSB            THIS INTEGER VARIABLE CONTAINS THE NUMBER OF SIZE
C                    FRACTIONS IN THE BREAKAGE SPECIFICATIONS.
C
      COMMON/FILES/READER,PRINTR,OFFLIN
      COMMON/WORK/NEXT,LAST,ICURVE,IUNIT,IDATA,ITITLE,ICHAR,IFLW
      COMMON/CODERC/ICOL,LCOL,LINE(40),CA(110)
      COMMON WS(6000)
      DIMENSION NCONS(9),CONS(5),ICONS(5),NCURVE(9)
      INTEGER WS,READER,PRINTR
      EQUIVALENCE (CONS(1),ICONS(1))
      DATA NCONS(1),NCONS(2),NCONS(3),NCONS(4)/0,0,0,1/
      DATA NCONS(5),NCONS(6),NCONS(7),NCONS(8)/1,1,5,3/
      DATA NCONS(9)/0/
      DATA NCURVE(1),NCURVE(2),NCURVE(3),NCURVE(4)/0,0,1,1/
      DATA NCURVE(5),NCURVE(6),NCURVE(7),NCURVE(8)/2,1,2,1/
      DATA NCURVE(9)/1/
    1 FORMAT(3I5,25X,40A1)
    2 FORMAT(16I5)
    3 FORMAT(10F8.0)
C
C     STEP 1         READ THE EQUIPMENT INVENTORY COUNT CARD
C
      READ(READER,1)NEQUIP
C
C     STEP 2         INITIALIZE AT ONE THE DUMMY COUNTER USED TO CONTROL
C                    THE MOVEMENT THROUGH THE INDIVIDUAL EQUIPMENT TYPES
```

FIGURE II-7-2.1. (Continued)

```
C                   WITHIN THE INVENTORY
C
      DO 70 IE=1,NEQUIP
C
C     STEP 3        READ THE EQUIPMENT TYPE SPECIFICATION CARD FOR THE
C                   CURRENT TYPE.
C
      READ(READER,1) ID,IUO,IBR,LINE
C
C     STEP 4        USING SUBROUTINE  R D D E S C  STORE THE EQUIPMENT
C                   TYPE DESCRIPTOR IN THE NEXT AVAILABLE CELLS OF
C                   WORKING STORAGE.
      ICOL=1
      LCOL=40
      IDESC=NEXT
      CALL RDDESC(WS(IDESC),NCELL)
      NEXT=NEXT+NCELL
C
C     STEP 5        STORE THE IDENTIFIER AND ENTER THE TYPE INTO THE
C                   STORAGE CHAIN.
C
      WS(NEXT)=ID
      NEXT=NEXT+1
      WS(NEXT)=0
      IF(IE.GT.1)GO TO 10
      ICURVE=NEXT
      GO TO 15
   10 WS(IEQUIP)=NEXT
   15 IEQUIP=NEXT
C
C     STEP 6        INITIALIZE THE MAIN SECTION OF THE EQUIPMENT TYPE
C                   STORAGE TABLE.
C
      NEXT=NEXT+5
      WS(IEQUIP+1)=IDESC
      WS(IEQUIP+2)=IUO
      WS(IEQUIP+3)=0
      WS(IEQUIP+4)=IBR
C
C     STEP 7        DETERMINE THE NUMBER OF CONSTANTS TO BE READ. IF
C                   ZERO, BRANCH TO STEP 9.
C
      NC=NCONS(IUO)
      IF(NC.EQ.0)GO TO 17
C
C     STEP 8        READ AND STORE THE CONSTANTS.
C
      READ(READER,3)(CONS(I),I=1,NC)
      DO 16 I=1,NC
```

FIGURE II-7-2.1. (Continued)

```
      WS(NEXT)=ICONS(I)
   16 NEXT=NEXT+1
C
C     STEP 9        DETERMINE THE NUMBER OF CURVE SETS TO BE READ. IF
C                   ZERO, BRANCH TO STEP 30.
C
   17 NC=NCURVE(IUO)
      IF(NC.EQ.0)GO TO 65
C
C     STEP 10       ALLOCATE SPACE IN WORKING STORAGE FOR THE CURVE SET
C                   INFORMATION CONTROL BLOCKS, AND RECORD THEIR ADDRESS.
C
      WS(IEQUIP+3)=NEXT
      ICV=NEXT-6
      NEXT=NEXT+6*NC
C
C     STEP 11       INITIALIZE AT ONE THE DUMMY COUNTER TO BE USED TO
C                   CONTROL THE MOVEMENT THROUGH THE CURVE SETS.
C
      DO 60 IC=1,NC
C
C     STEP 12       UPDATE THE CURVE SET INFORMATION CONTROL BLOCK
C                   ADDRESS SO THAT IT POINTS TO THE BLOCK FOR THE
C                   CURRENT CURVE SET.
C
      ICV=ICV+6
C
C     STEP 13       READ AND STORE THE MAIN CURVE CONTROL INFORMATION.
C
      READ(READFF,2)ITYPE,NPOINT,NSIZE
      WS(ICV)=ITYPE
      WS(ICV+1)=NPOINT
      WS(ICV+2)=NSIZE
C
C     STEP 14       IF SIMPLE VECTORS ARE BEING ENTERED, BRANCH TO
C                   STEP 23.
C
      IF(ITYPE.EQ.1)GO TO 35
C
C     STEP 15       IF SIMPLE BIVARIATE DISTRIBUTIONS CURVES ARE BEING
C                   READ, BRANCH TO STEP 25.
C
      IF(ITYPE.EQ.2)GO TO 45
C
C     STEP 16       THE CURVE SET HAS A SIZE CLASSIFICATION SCHEME ASSO-
C                   CIATED WITH IT. RECORD THE ADDRESS IN WORKING STOR-
C                   AGE WHERE THE SIZE DISTRIBUTION IS TO BE STORED, AND
C                   THE READ AND STORE IT.
C
      WS(ICV+5)=NEXT
```

FIGURE II-7-2.1. (Continued)

```
      LCELL=NEXT+NSIZE
      READ(READER,3)(WS(I),I=NEXT,LCELL)
      NEXT=LCELL+1
C
C     STEP 17       IF A CLASSIFICATION CURVE IS BEING READ, BRANCH TO
C                   STEP 22.
C
      IF(ITYPE.EQ.3)GO TO 30
C
C     STEP 18       IF A C-GENERALIZED DISTRIBUTION CURVE IS BEING READ,
C                   BRANCH TO STEP 25.
C
      IF(ITYPE.EQ.4)GO TO 45
C
C     STEP 19       EXPLICIT DISTRIBUTION CURVES ARE BEING READ. READ
C                   AND STORE THE NUMBER OF CURVES PER SIZE FRACTION.
C
      LCELL=NEXT+NSIZE-1
      READ(READER,2)(WS(I),I=NEXT,LCELL)
C
C     STEP 20       COMPUTE THE TOTAL NUMBER OF CURVES TO BE READ.
C
      NCRV=0
      DO 25 I=NEXT,LCELL
      NCRV=NCRV+WS(I)
   25 CONTINUE
      NEXT=LCELL+1
      LCELL=NEXT+NCRV-1
C
C     STEP 21       READ AND STORE THE DECISION VARIABLE VALUES ASSO-
C                   CIATED WITH EACH OF THE EXPLICIT CURVES. THEN BRANCH
C                   TO STEP 26.
C
      READ(READER,3)(WS(I),I=NEXT,LCELL)
      NEXT=LCELL+1
      GO TO 50
C
C     STEP 22       SET THE NUMBER OF CURVES EQUAL TO THE NUMBER OF
C                   POINTS, AND THE NUMBER OF POINTS EQUAL TO THE
C                   NUMBER OF SIZE FRACTIONS. THEN BRANCH TO STEP 24.
C
   30 NCRV=NPOINT
      NPOINT=NSIZE
      GO TO 40
C
C     STEP 23       SET THE NUMBER OF CURVES EQUAL TO THE NUMBER OF SIZE
C                   FRACTIONS.
C
   35 NCRV=NSIZE
```

FIGURE II-7-2.1. (Continued)

```
C
C     STEP 24      INITIALIZE THE ADDRESS OF THE DEPENDENT VALUES AT
C                  ZERO, AND BRANCH TO STEP 27.
C
   40 WS(ICV+4)=0
      GO TO 55
C
C     STEP 25      SET THE NUMBER OF CURVES EQUAL TO THE NUMBER OF SIZE
C                  FRACTIONS.
C
   45 NCRV=NSIZE
C
C     STEP 26      STORE THE ADDRESS OF AND ALLOCATE SPACE FOR THE DE-
C                  PENDENT VALUES.
C
   50 WS(ICV+4)=NEXT
      NEXT=NEXT+NPOINT*NCRV
C
C     STEP 27      STORE THE ADDRESS OF AND ALLOCATE SPACE FOR THE IN-
C                  DEPENDENT VALUES.
C
   55 WS(ICV+3)=NEXT
      NEXT=NEXT+NPOINT*NCRV
C
C     STEP 28      READ AND STORE THE ACTUAL CURVE VALUES.
C
      IX=WS(ICV+3)
      IY=WS(ICV+4)
      DO 56 ICS=1,NCRV
      LCELL=IX+NPOINT-1
      READ(READER,3)(WS(I),I=IX,LCELL)
      IX=LCELL+1
      IF(IY.EQ.0)GO TO 56
      LCELL=IY+NPOINT-1
      READ(READER,3)(WS(I),I=IY,LCELL)
      IY=LCELL+1
   56 CONTINUE
C
C     STEP 29      INCREMENT BY ONE THE DUMMY COUNTER CONTROLLING THE
C                  MOVEMENT THROUGH THE CURVE SETS FOR THE CURRENT
C                  EQUIPMENT TYPE, AND IF ALL HAVE NOT BEEN PROCESSED,
C                  LOOP BACK TO STEP 12.
C
   60 CONTINUE
   65 CONTINUE
C
C     STEP 30      IF THERE IS NO BREAKAGE FOR THIS UNIT, BRANCH TO
C                  STEP 35.
C
```

FIGURE II-7-2.1. (Continued)

```
      IF(IBR.EQ.0) GOTO 70
C
C     STEP 31       READ AND STORE THE NUMBER OF SIZE FRACTIONS DEFINING
C                   THE BREAKAGE FOR THE UNIT.
C
      READ(READER,1) NSB
      WS(IEQUIP+4)=NEXT
      WS(NEXT)=NSB
      NEXT=NEXT+1
C
C     STEP 32       READ AND STORE THE POWER OF THE BREAKAGE MATRIX TO
C                   BE USED FOR THIS UNIT
C
      READ(READER,1) WS(NEXT)
      NEXT=NEXT+1
C
C     STEP 33       READ AND STORE THE SIZE FRACTIONS DEFINING THE
C                   BREAKAGE FOR THE UNIT.
C
      LCELL=NEXT+NSB
      READ(READER,3) (WS(I),I=NEXT,LCELL)
      NEXT=LCELL+1
C
C     STEP 34       READ AND STORE THE SELECTION FACTORS FOR BREAKAGE.
C
      LCELL=NEXT+NSB-1
      READ(READER,3) (WS(I),I=NEXT,LCELL)
      NEXT=LCELL+1
C
C     STEP 35       INCREMENT BY 1 THE DUMMY COUNTER CONTROLLING THE
C                   MOVEMENT THROUGH THE EQUIPMENT TYPES, AND IF ALL
C                   HAVE NOT BEEN PROCESSED, LOOP BACK TO STEP 3.
C
   70 CONTINUE
C
C     STEP 36       END LOCAL PROCESSING.
C
      RETURN
      END
```

FIGURE II-7-2.1. (Continued)

7.3 Subroutine RDFEED

Subroutine RDFEED reads the descriptions of the input flows. Figure II-7-3.1 contains the listing and basic documentation.

Reproduced from best available copy.

```
      SUBROUTINE RDFEED(TEMP,NSIZE,NGRAV,NCHAR)
C
C     TITLE(READ FEED DESCRIPTION FOR SAVING ON THE DATA BASE)
C
C     SUBROUTINE RDFEED IS USED TO READ THE VALUES FOR THE EXTERNAL
C     FEED STREAMS ENTERING INTO A PLANT.
C
C     PARAMETER      DESCRIPTION
C     TEMP           THIS REAL 3-DIMENSIONAL ARRAY RETURNS THE VALUE
C                    OF THE COMPOSIT FEED ENTERING THE PLANT.
C     NSIZE          THIS INTEGER VARIABLE CONTAINS THE NUMBER OF SIZE
C                    INCREMENTS.
C     NGRAV          THIS INTEGER VARIABLE CONTAINS THE NUMBER OF SPEC-
C                    IFIC GRAVITY INCREMENTS.
C     NCHAR          THIS INTEGER VARIABLE CONTAINS THE NUMBER OF
C                    CHARACTERISTICS.
C
C     COMMON         VARIABLES USED
C     /WORK/         IBTU,IASH
C     /FILES/        READER
C
C     EXTERNALS      YINTRP
C
C     VARIABLE       DESCRIPTION
C     WT             THIS REAL 1-DIMENSIONAL ARRAY IS USED TO CONTAIN THE
C                    WEIGHTS FOR THE SIZE INCREMENTS FOR THE CURRENT
C                    FEED.
C     TBTU           THIS REAL 1-DIMENSIONAL ARRAY CONTAINS THE BTUS PER
C                    POUND BY SPECIFIC GRAVITY GRAVITY INCREMENT FOR THE
C                    CURRENT FEED.
C     I              THIS DUMMY VARIABLE IS USED AS A COUNTER TO CONTROL
C                    THE MOVEMENT THROUGH THE SIZE INCREMENTS.
C     J              THIS DUMMY VARIABLE IS USED AS A COUNTER TO CONTROL
C                    THE MOVEMENT THROUGH THE SPECIFIC GRAVITY INCREMENTS.
C     K              THIS DUMMY VARIABLE IS USED AS A COUNTER TO CONTROL
C                    THE MOVEMENT THROUGH THE FEED CLASSIFICATION VALUES.
C     JADJ           THIS INTEGER VARIABLE CONTAINS THE ADJUSTMENT CODE
C                    WHICH INDICATES WHETHER OR NOT THE FEED DESCRIPTIONS
C                    SHOULD BE ADJUSTED TO INCLUDE AN ESTIMATED SMALLEST
C                    SIZE FRACTION.
C     SUM1           THIS REAL VARIABLE CONTAINS THE TOTAL WEIGHT AC-
C                    COUNTED FOR BY THE INPUT WEIGHTS.
C     COM            THIS REAL VARIABLE CONTAINS THE AMOUNT OF SOME
C                    CHARACTERISTIC IN THE SMALLEST SIZE FRACTION IF THE
C                    DISTRIBUTION OF THAT MATERIAL IN THAT FRACTION WERE
C                    THE SAME AS IN THE NEXT TO SMALLEST FRACTION.
C     JBTU           THIS INTEGER VARIABLE CONTAINS THE BTU VALUE CODE
C                    WHICH INDICATES THE LEVEL OF DETAIL TO WHICH BTU
C                    VALUES ARE SPECIFIED.
```

FIGURE II-7-3.1. LISTING FOR SUBROUTINE RDFEED

```
C     NS              THIS INTEGER VARIABLE CONTAINS THE NUMBER OF SIZE
C                     FRACTIONS FOR WHICH DETAILED DATA IS BEING READ.
C     SUM2            THIS REAL VARIABLE CONTAINS THE WEIGHT OF MATERIAL
C                     ACCOUNTED FOR IN A PARTICULAR SIZE FRACTION.
C     VAL             THIS REAL 1-DIMENSIONAL ARRAY CONTAINS THE OVERALL
C                     CHARACTERISTIC VALUES FOR THE SMALLEST SIZE FRACTION.
C     NPOINT          THIS INTEGER VARIABLE CONTAINS THE NUMBER OF POINTS
C                     USED TO DEFINE THE PERCENT ASH - BTU VALUE FUNCTION.
C     VSLBTU          THIS 1-DIMENSIONAL REAL ARRAY CONTAINS THE BTU VALUES
C                     FOR INTERPOLATION.
C     VSLYLD          THIS 1-DIMENSIONAL REAL ARRAY CONTAINS THE PERCENT
C                     ASH VALUES FOR INTERPOLATION.
C     NC              THIS INTEGER VARIABLE CONTAINS ONE LESS THAN THE
C                     NUMBER OF CHARACTERISTICS.
C
      COMMON/WORK/NEXT,LAST,ICURVE,IUNIT,IDATA,ITITLE,ICHAR,IFLW,IASH,
     1 ISUL,IBTU
      DIMENSION TEMP(NSIZE,NGRAV,NCHAR)
      DIMENSION VSLYLD(60),VSLBTU(60)
      COMMON/FILES/READER,PRINTR,OFFLIN
      DIMENSION WT(23),TBTU(20),VAL(50)
      INTEGER READER,PRINTR
    1 FORMAT(13F6.3)
    2 FORMAT(8F10.5)
    3 FORMAT(1I5)
    4 FORMAT(10X,I3,3X,I3)
C
C     STEP 1          READ THE CONTROL DATA FOR THE FEED
C
      READ(READER,4) JADJ,JBTU
      NS=NSIZE
      IF(JADJ.EQ.1)NS=NS-1
C
C     STEP 2          READ THE WEIGHTS ASSOCIATED WITH EACH SIZE INCRE-
C                     MENT FOR THE CURRENT FEED.
C
      READ(READER,1)(WT(I),I=1,NSIZE)
C
C     STEP 3          READ THE BTUS PER SPECIFIC GRAVITY INCREMENT FOR
C                     THE CURRENT FEED.
C
      IF(JBTU.EQ.0)READ(READER,2)(TBTU(J),J=1,NGRAV)
C
C     STEP 4          READ THE VALUES FOR THE CURRENT FEED, NORMALIZE IT,
C                     AND CONVERT THE PERCENTAGES TO RATIOS.
C
      SUM1=0.0
      DO 17 I=1,NS
      SUM1=SUM1+WT(I)
```

FIGURE II-7-3.1. (Continued)

```
      SUM2=0.0
      DO 15 J=1,NGRAV
      READ(READER,2)(TEMP(I,J,K),K=1,NCHAR)
      IF(JBTU.EQ.0) TEMP(I,J,IBTU)=TBTU(J)
      SUM2=SUM2+TEMP(I,J,1)
      DO 15 K=2,NCHAR
      IF(K.EQ.IBTU) GOTO 15
      TEMP(I,J,K)=TEMP(I,J,K)/100.0
   15 CONTINUE
      IF(SUM2.EQ.0.0)GO TO 17
      DO 16 J=1,NGRAV
      TEMP(I,J,1)=TEMP(I,J,1)/SUM2
   16 CONTINUE
   17 CONTINUE
      IF(JADJ.NE.1)GO TO 25
      SUM1=SUM1+WT(NSIZE)
      NC=NCHAR-1
      READ(READER,2) (VAL(K),K=1,NC)
      DO 18 J=1,NGRAV
      TEMP(NSIZE,J,1)=TEMP(NS,J,1)
      TEMP(NSIZE,J,IBTU)=TEMP(NS,J,IBTU)
   18 CONTINUE
      DO 21 K=2,NCHAR
      IF(K.EQ.IBTU) GOTO 21
      COM=0.0
      DO 19 J=1,NGRAV
      TEMP(NSIZE,J,K)=TEMP(NS,J,K)
      COM=COM+TEMP(NSIZE,J,1)*TEMP(NSIZE,J,K)
   19 CONTINUE
      IF(COM.EQ.0.0 .OR. VAL(K-1).EQ.0.0)GO TO 21
      COM=COM/VAL(K-1)*100.0
      DO 20 J=1,NGRAV
      TEMP(NSIZE,J,K)=TEMP(NSIZE,J,K)/COM
   20 CONTINUE
   21 CONTINUE
   25 DO 30 I=1,NSIZE
      WT(I)=WT(I)/SUM1
      DO 30 J=1,NGRAV
      TEMP(I,J,1)=TEMP(I,J,1)*WT(I)
   30 CONTINUE
      IF(JBTU.NE.2)GO TO 36
      READ(READER,3)NPOINT
      READ(READER,2)(VSLYLD(I),I=1,NPOINT)
      READ(READER,2)(VSLBTU(I),I=1,NPOINT)
      DO 35 I=1,NSIZE
      DO 35 J=1,NGRAV
      COM=TEMP(I,J,IASH)*100.0
      TEMP(I,J,IBTU)=YINTRP(VSLYLD,VSLBTU,COM,NPOINT)
   35 CONTINUE
```

FIGURE II-7-3.1. (Continued)

```
   36 CONTINUE
      DO 37 I=1,NSIZE
      DO 37 J=1,NGRAV
      DO 37 K=2,NCHAR
   37 TEMP(I,J,K)=TEMP(I,J,K)*TEMP(I,J,1)
C
C     STEP 5        END LOCAL PROCESSING.
C
      RETURN
      END
```

FIGURE II-7-3.1. (Continued)

7.4 Subroutine CONFSV

Subroutine CONFSV is used to create a data base of the configuration in the case where the input data to the program is read from cards. The configuration data base, consisting of the equipment inventory as a header record followed by up to thirty plant configurations distinguished by a four-letter key, is created on file number 3 and may be saved at the end of the program run for permanent file storage of the configurations. Figure II-7-4.1 contains the listing and basic documentation of subroutine CONFSV.

```
      SUBROUTINE CONFSV
C
C     TITLE(CREATE A CONFIGURATION DATA BASE)
C
C     SUBROUTINE CONFSV CREATES A DATA BASE OF PLANT CONFIGURATIONS.
C
C     COMMON        VARIABLES USED
C     /CNTVAR/      NUNITS,NFLOWS
C     /FILES/       READER
C     /WORK/        NEXT,ICURVE,IUNIT
C     //            SG
C
C     EXTERNALS SVBASE,RDCURV,RDUNIT
C
C     VARIABLE      DESCRIPTION
C     NREC          THIS INTEGER VARIABLE CONTAINS THE LENGTH OF THE
C                   HEADER RECORD FOR THE FILE.
C     FILE          THIS ONE-DIMENSIONAL INTEGER ARRAY CONTROLS THE
C                   WRITING OF THE DATA BASE.
C     KEY           THIS INTEGER VARIABLE IS USED TO CONTAIN THE KEYS
C                   FOR THE PLANT CONFIGURATIONS ON THE DATA BASE.
C     BLANK         THIS INTEGER VARIABLE CONTAINS A BLANK KEY.
C     NST,NFIN      THESE INTEGER VARIABLES ARE USED AS POINTERS IN THE
C                   HEADER RECORD.
C     I             THIS INTEGER VARIABLE IS USED AS A DO-LOOP COUNTER.
C
      COMMON/CNTVAR/KF(2,100),SIZE(23),GBOUND(21),GRAV(20)
     1,WFLOW(100) ,S(100),D1(60),D2(60),D3(60),L(60)
     2                          ,NSIZE,NGRAV,IOUT,NUNITS,NFLOWS,NCHAR
      COMMON/FILES/READER,PRINTR,OFFLIN
      COMMON/WORK/NEXT,LAST,ICURVE,IUNIT,IDATA,ITITLE,ICHAR,IFLW
      COMMON SG(6000)
      DIMENSION FILE(2)
      INTEGER FILE,READER,PRINTR,BLANK
      INTEGER SG
      DATA FILE(1)/3/,FILE(2)/0/,BLANK/4H    /
C
C     STEP 1        READ THE HEADER INFORMATION FOR THE FILE - THE
C                   EQUIPMENT INVENTORY
C
      NEXT=1
      CALL RDCURV
      SG(NEXT)=ICURVE
      NREC=NEXT
C
C     STEP 2        OPEN THE SAVE FILE AND WRITE THE HEADER INFORMATION
C
      CALL SVBASE(FILE,SG,NREC,0)
C
```

FIGURE II-7-4.1. LISTING FOR SUBROUTINE CONFSV

```
C     STEP 3          READ THE KEY FOR THE NEXT CONFIGURATION.  IF ZERO
C                     THERE ARE NO MORE CONFIGURATIONS, BRANCH TO STEP 7
C
   10 READ(READER,11) KEY
   11 FORMAT(A4)
      IF(KEY.EQ.BLANK) GOTO 30
      READ(READER,21) NUNITS,NFLOWS
      NST=NEXT
   21 FORMAT(2I5)
C
C     STEP 4          READ THE UNIT DESCRIPTORS
C
      CALL RDUNIT
C
C     STEP 5          READ THE FLOW DESCRIPTION
C
      NFIN=NEXT+3*NFLOWS-1
      READ(READER,31) (SG(I),I=NEXT,NFIN)
   31 FORMAT(2I5,1X,A1)
      NEXT=NFIN+1
      SG(NEXT)=NUNITS
      NEXT=NEXT+1
      SG(NEXT)=NFLOWS
      NEXT=NEXT+1
      SG(NEXT)=IUNIT
      NFIN=NEXT
      NREC=NFIN-NST+1
C
C     STEP 6          SAVE ON THE SAVE FILE
C
      CALL SVBASE(FILE,SG(NST ),NREC,KEY)
      GOTO 10
   30 CONTINUE
C
C     STEP 7          CLOSE THE SAVE FILE AND END LOCAL PROCESSING
C
      CALL SVBASE(FILE,SG,NREC,0)
      RETURN
      END
```

FIGURE II-7-4.1. (Continued)

7.5 Subroutine FEEDSV

Subroutine FEEDSV is used to create a data base of the input feed descriptions in the case where the input data to the program is read from cards. The feed data base, consisting of the size fractions, specific gravity fractions and characteristics as a header record followed by up to thirty feed descriptions each distinguished by a four-letter key, is created on file number 4 and may be saved at the end of the program run for permanent file storage of the feeds. Figure II-7-5.1 contains the listing and basic documentation of subroutine FEEDSV.

Reproduced from
best available copy.

```
      SUBROUTINE FEEDSV
C
C     TITLE(CREATE A FEED DATA BASE)
C
C     SUBROUTINE FEEDSV CREATES A DATA BASE OF FEEDS.
C
C     COMMON        VARIABLES USED
C     /WORK/        IASH,ISUL,IBTU
C     /CODERC/      ICOL,LCOL,LINE
C     /FILES/       READER
C     /CHRSET/      LETTER
C     //            REC,IREC
C
C     EXTERNALS     RDFEED,SVBASE
C
C     VARIABLE      DESCRIPTION
C     NSIZE         THIS INTEGER VARIABLE CONTAINS THE NUMBER OF SIZE
C                   FRACTIONS OF THE FEED.
C     NGRAV         THIS INTEGER VARIABLE CONTAINS THE NUMBER OF
C                   SPECIFIC GRAVITY FRACTIONS FOR THE FEED.
C     NCHAR         THIS INTEGER VARIABLE CONTAINS THE NUMBER OF
C                   CHARACTERISTICS FOR THE FEED.
C     NREC          THIS INTEGER VARIABLE CONTAINS THE LENGTH OF THE
C                   HEADER RECORD FOR THE DATA BASE.
C     NST,NFIN      THESE INTEGER VARIABLES ARE USED AS POINTERS
C                   IN THE HEADER RECORD.
C     NVAL          THIS INTEGER VARIABLE IS USED TO CONTAIN THE
C                   VALUE OF GROUPS OF CHARACTERS IN A DESCRIPTOR.
C     FILE          THIS 1-DIMENSIONAL INTEGER ARRAY CONTROLS THE
C                   WRITING OF THE DATA BASE.
C     KEY           THIS INTEGER VARIABLE IS USED TO CONTAIN THE KEYS FOR
C                   THE FEEDS ON THE DATA BASE.
C     BLANK         THIS INTEGER VARIABLE CONTAINS A BLANK KEY.
C     I,J,K         THESE INTEGER VARIABLES ARE USED AS DO-LOOP COUNTERS.
C
      COMMON/CODERC/ICOL,LCOL,LINE(150)
      COMMON/FILES/READER,PRINTR
      COMMON/CHRSET/LETTER(47)
      COMMON/WORK/NEXT,LAST,ICURVE,IUNIT,IDATA,ITITLE,ICHAR,IFLW,IASH,
     1 ISUL,IBTU
      COMMON REC(6000)
      DIMENSION IREC(6000)
      EQUIVALENCE (REC(1),IREC(1))
      DIMENSION FILE(2)
      INTEGER FILE,READER,PRINTR,BLANK
      DATA FILE(1)/4/,FILE(2)/0/,BLANK/4H    /
C
C     STEP 1        READ THE HEADER INFORMATION FOR THE FILE - THE SIZE
C                   FRACTIONS, SPECIFIC GRAVITY FRACTIONS, AND
```

FIGURE II-7-5.1. LISTING FOR SUBROUTINE FEEDSV

```
C                    CHARACTERISTICS.
C
      READ(READER,11) NSIZE,NGRAV,NCHAR
   11 FORMAT(3I5)
      REC(1)=NSIZE
      REC(2)=NGRAV
      REC(3)=NCHAR
      NREC=3
      NST=NREC+1
      NFIN=NREC+NSIZE+1
      READ(READER,21) (REC(I),I=NST,NFIN)
      NREC=NFIN
      NST=NREC+1
      NFIN=NREC+NGRAV+1
      READ(READER,21) (REC(I),I=NST,NFIN)
      NREC=NFIN
      NST=NREC+1
      NFIN=NREC+3
      READ(READER,11) (REC(I),I=NST,NFIN)
      NREC=NFIN
      IASH=IREC(NST)
      ISUL=IREC(NST+1)
      IBTU=IREC(NST+2)
   21 FORMAT(10F8.0)
      NST=NREC+1
      DO 40 J=1,NCHAR
      READ(READER,41) (LINE(I),I=1,49)
      DO 25 I=1,49
      DO 23 K=1,47
      IF(LINE(I)-LETTER(K)) 23,24,23
   23 CONTINUE
   24 LINE(I)=K
   25 CONTINUE
      NVAL=0
      DO 26 I=1,5
   26 NVAL=NVAL*64+LINE(I)
      IREC(NST)=5
      IREC(NST+1)=NVAL
      NST=NST+2
      IREC(NST)=30
      ICOL=6
      DO 28 I=1,6
      LCOL=ICOL+4
      NVAL=0
      DO 27 K=ICOL,LCOL
   27 NVAL=NVAL*64+LINE(K)
      NST=NST+1
      IREC(NST)=NVAL
   28 ICOL=LCOL+1
```

FIGURE II-7-5.1. (Continued)

```
      NST=NST+1
      IREC(NST)=7
      NVAL=0
      DO 29 I=36,40
   29 NVAL=NVAL*64+LINE(I)
      IREC(NST+1)=NVAL
      IREC(NST+2)=LINE(41)*64+LINE(42)
      NST=NST+3
      IREC(NST)=7
      NVAL=0
      DO 32 I=43,47
   32 NVAL=NVAL*64+LINE(I)
      IREC(NST+1)=NVAL
      IREC(NST+2)=LINE(48)*64+LINE(49)
   40 NST=NST+3
      NREC=NST-1
   41 FORMAT(5A1,1X,30A1,1X,7A1,1X,7A1)
C
C     STEP 2        OPEN THE SAVE FILE AND WRITE THE HEADER INFORMATION
C
      CALL SVBASE(FILE,REC,NREC,0)
      NREC=NSIZE*NGRAV*NCHAR
C
C     STEP 3        READ THE KEY FOR THE NEXT FEED.  IF ZERO THERE
C                   ARE NO MORE FEEDS, BRANCH TO STEP 5.
C
   20 READ(READER,31) KEY
   31 FORMAT(A4)
      IF(KEY.EQ.BLANK) GOTO 30
C
C     STEP 4        READ THE FEED DESCRIPTION AND SAVE ON THE SAVE FILE
C
      CALL RDFEED(REC,NSIZE,NGRAV,NCHAR)
      CALL SVBASE(FILE,REC,NREC,KEY)
      GOTO 20
C
C     STEP 5        CLOSE THE SAVE FILE.  END PROCESSING.
C
   30 CALL SVBASE(FILE,REC,NREC,0)
      RETURN
      END
```

FIGURE II-7-5.1. (Continued)

7.6 Subroutine RDCFBS

Subroutine RDCFBS reads a configuration, selected according to its key, from the configuration data base. Figure II-7-6.1 contains the listing and basic documentation for subroutine RDCFBS.

```
      SUBROUTINE RDCFBS(FILE3)
C
C     TITLE(READ A CONFIGURATION FROM THE DATA BASE)
C
C     SUBROUTINE RDCFBS SELECTS AND DISPLAYS THE DESIRED PLANT
C     CONFIGURATION FROM THE DATA BASE.
C
C     PARAMETER     DESCRIPTION
C     FILE3         THIS 1-DIMENSIONAL INTEGER ARRAY CONTROLS THE
C                   READING OF THE DATA BASE.
C
C     COMMON        VARIABLES USED
C     /CODERC/      ICOL
C     /FILES/       READER
C     /CHRSET/      LETTER
C     /CNTVAR/      NFLOWS,NUNITS,D1,D2,D3,KF,L,S
C     /WORK/        NEXT,IUNIT
C     //            SG
C     /IOCNT/       JOB,CARCON
C
C     EXTERNALS     RDBASE,PRHEAD,PRDESC,CNVTIE,PPLINE,VESSEL
C
C     VARIABLE      DESCRIPTION
C     LD1-LD4       THESE ONE-DIMENSIONAL INTEGER ARRAYS CONTAIN
C                   PACKED CHARACTER CODES FOR OUTPUT PURPOSES.
C     KEY           THIS INTGER VARIABLE CONTAINS THE KEY FOR THE
C                   REQUIRED CONFIGURATION.
C     NST,NFIN      THESE INTEGER VARIABLES ARE USED AS POINTERS IN
C     IOLD,NU       WORKING STORAGE.
C
C     NREC          THIS INTEGER VARIABLE CONTAINS THE LENGTH OF THE
C                   DATA BASE RECORD.
C     I,J,U         THESE INTEGER VARIABLES ARE USED AS DUMMY COUNTERS.
C     LINES         THIS INTEGER VARIABLE IS USED AS A COUNTER FOR THE
C                   NUMBER OF LINES
C     ID            THIS INTEGER VARIABLE IS THE EQUIPMENT IDENTIFIER.
C     IUO           THIS INTEGER VARIABLE CONTAINS THE UNIT OPERATION
C                   TYPE CODE.
C     IEQUIP        THIS INTEGER VARIABLE CONTAINS THE ADDRESS IN
C                   WORKING STORAGE OF THE EQUIPMENT INFORMATION FOR THE
C                   UNIT.
C
      COMMON/CODERC/ICOL,LCOL,LINE(150)
      COMMON/FILES/READER,PRINTR,OFFLIN
      COMMON/CHRSET/LETTER(47)
      COMMON/CNTVAR/KF(2,100),SIZE(23),GBOUND(21),GRAV(20),WFLOW(100)
     1             ,S(100),D1(60),D2(60),D3(60),L(60)
     2                           ,NSIZE,NGRAV,IOUT,NUNITS,NFLOWS,NCHAR
      COMMON/WORK/NEXT,LAST,ICURVE,IUNIT,IDATA,ITITLE,ICHAR,IFLW
```

FIGURE II-7-6.1. LISTING FOR SUBROUTINE RDCFBS

```
      COMMON SG(6000)
      COMMON/IOCNT/JOB,CARCON
      INTEGER CARCON
      INTEGER READER,PRINTR,OFFLIN,S
      INTEGER J
      INTEGER SG
      INTEGER FILE3,D1,D2,D3
      DIMENSION LD1(4),LD2(3),LD3(5),LD4(10)
      DATA LD1(1),LD1(2),LD1(3),LD1(4)/11,356029741,240439429,18/
      DATA LD2(1),LD2(2),LD2(3)/9,356029741,5346309/
      DATA LD3(1),LD3(2),LD3(3),LD3(4)/18,66432467,154987350,21532738/
      DATA LD3(5)/49491/
      DATA LD4(1),LD4(2),LD4(3),LD4(4)/44,154755148,152228549,431252434/
      DATA LD4(5),LD4(6),LD4(7)/755823494,152917121,337965997/
      DATA LD4(8),LD4(9),LD4(10)/67453037,33894767,12516335/
C     STEP 1        READ THE KEY FOR THE REQUIRED CONFIGURATION.
      READ(READER,31) KEY
   31 FORMAT(A4)
C     STEP 2        SET THE POINTERS IN WORKING STORAGE AND READ THE
C                   PLANT CONFIGURATION CORRESPONDING TO THE KEY.
      NST=NEXT-IUNIT
      IOLD=IUNIT
      IUNIT=NEXT
      CALL RDBASE(FILE3,SG(NEXT),NREC,KEY)
      IF(NREC.EQ.0) GOTO 999
C     STEP 3        UNPACK THE DETAILS OF THE CONFIGURATION READ FROM
C                   THE DATA BASE.
      NFIN=NEXT+NREC-1
      NEXT=NEXT+SG(NFIN)
      NFIN=NFIN-1
      NFLOWS=SG(NFIN)
      NFIN=NFIN-1
      NUNITS=SG(NFIN)
      NU=IUNIT+NUNITS-1
      DO 12 I=IUNIT,NU
      IF(SG(I).GT.IOLD) SG(I)=SG(I)+NST
   12 CONTINUE
      I=NFIN-3*NFLOWS
      J=1
   15 KF(1,J)=SG(I)
      KF(2,J)=SG(I+1)
      S(J)=SG(I+2)
      J=J+1
      I=I+3
      IF(J.LE.NFLOWS) GOTO 15
      I=NEXT
      U=1
   16 L(U)=SG(I)
      D1(U)=SG(I+1)
```

FIGURE II-7-6.1. (Continued)

```
      D2(U)=SG(I+2)
      D3(U)=SG(I+3)
      U=U+1
      I=I+4
      IF(U.LE.NUNITS) GOTO 16
C     STEP 4        PRINT OUT THE DETAILS OF THE CONFIGURATION.
      LINES=52
      DO 80 U=1,NUNITS
      IEQUIP=L(U)
      ID=SG(IEQUIP-1)
      IUO=SG(IEQUIP+2)
      IF(LINES.LT.51) GOTO 60
      LINES=0
      CALL PRHEAD
      JOB=-1
      CARCON=LETTER(45)
      ICOL=10
      CALL PRDESC(LD1)
      ICOL=ICOL+24
      CALL PRDESC(LD2)
      ICOL=ICOL+24
      CALL PRDESC(LD3)
      CALL PRLINE
   60 LINES=LINES+2
      CARCON=LETTER(45)
      CALL PRLINE
      ICOL=14
      CALL CNVTIE(U,2,-1)
      ICOL=ICOL+23
      CALL CNVTIE(ID,2,-1)
      CALL PRLINE
      IF(IUO.EQ.6.OR.IUO.EQ.1.OR.IUO.EQ.9) GOTO 75
      IF(IUO.EQ.7) GOTO 70
      CARCON=LETTER(37)
      ICOL=84
      CALL CNVTIE(D1(U),7,3)
      CALL PRLINE
      GOTO 75
   70 CARCON=LETTER(37)
      ICOL=73
      CALL CNVTIE(D1(U),8,3)
      CALL CNVTIE(D2(U),8,3)
      CALL CNVTIE(D3(U),8,3)
      CALL PRLINE
   75 CALL VESSEL(U,1)
   80 CONTINUE
C     STEP 5        BRANCH TO STEP 7.
      GOTO 900
C     STEP 6        THERE IS NO CONFIGURATION CORRESPONDING TO THE
```

FIGURE II-7-6.1. (Continued)

```
C                    GIVEN KEY.  PRINT OUT AN ERROR MESSAGE.
  999 ICOL=0
      CALL PRDESC(LD4)
      CALL PRLINE
C     STEP 7         END LOCAL PROCESSING.
  900 CONTINUE
      RETURN
      END
```

FIGURE II-7-6.1. (Continued)

7.7 Subroutine RDFDBS

Subroutine RDFDBS reads the header record from the feed data base and then reads the input feeds to the plant, selected from the data base by their keys. Figure II-7-7.1 contains the listing and basic documentation for subroutine RDFDBS.

```
      SUBROUTINE RDFDBS(NFEED,FLOWT)
C
C     TITLE(READ THE FEED DESCRIPTION FROM THE DATA BASE)
C
C     SUBROUTINE RDFDBS READS THE SELECTED INPUT FEEDS FROM THE DATABASE
C     AND COMBINES THEM INTO A SINGLE INPUT FEED TO THE PLANT.
C
C     PARAMETERS   DESCRIPTION
C     NFEED        THIS INTEGER VARIABLE RETURNS THE NUMBER OF FEEDS TO
C                  THE PLANT.
C     FLOWT        THIS REAL VARIABLE RETURNS THE TOTAL RATE OF FLOW OF
C                  THE INPUT FEED TO THE PLANT.
C
C     COMMON       VARIABLES USED
C     /CODERD/     ICOL
C     /WORK/       NEXT,IFLW,IDATA,ICHAR,IASH,ISUL,IBTU
C     /CNTVAR/     GBOUND,GRAV,NCHAR,NFLOWS,NGRAV,NSIZE,S,SIZE
C     /BLK1/       FILE4,IREC,REC
C     /FILES/      READER
C     //           SG,ISG
C
C     EXTERNALS    OUTPT1,PRDESC,PRLINE,RDBASE
C
C     VARIABLES    DESCRIPTION
C     I,J,K        THESE INTEGER VARIABLES ARE USED AS DO-LOOP COUNTERS.
C     II,JJ,IT1,   THESE INTEGER VARIABLES ARE USED AS ARRAY
C     IT2          POINTERS.
C     KEY          THIS INTEGER VARIABLE IS USED TO CONTAIN THE KEYS FOR
C                  THE FEEDS ON THE DATA BASE.
C     LD1          THIS 1-DIMENSIONAL INTEGER ARRAY CONTAINS PACKED
C                  CHARACTER CODES FOR DISPLAY.
C     NG           THIS INTEGER VARIABLE CONTAINS ONE PLUS THE NUMBER
C                  OF SPECIFIC GRAVITY FRACTIONS.
C     NS           THIS INTEGER VARIABLE CONTAINS ONE PLUS THE NUMBER
C                  OF SIZE FRACTIONS.
C     NREC         THIS INTEGER VARIABLE CONTAINS THE LENGTH OF THE
C                  RECORD READ FROM THE DATA BASE.
C     TFLOW        THIS REAL VARIABLE CONTAINS THE FLOWRATE OF THE
C                  INDIVIDUAL FEEDS.
C
      COMMON/CODERD/ICOL,LCOL,LINE(150)
      COMMON/WORK/NEXT,LAST,ICURVE,IUNIT,IDATA,ITITLE,ICHAR,IFLW,IASH,
     1 ISUL,IBTU
      COMMON/CNTVAR/KF(2,100),SIZE(23),GBOUND(21),GRAV(20),WFLOW(100)
     1             ,S(100),D1(60),D2(60),D3(60),L(60)
     2                          ,NSIZE,NGRAV,IOUT,NUNITS,NFLOWS,NCHAR
      COMMON/FILES/READER,PRINTR,OFFLIN
      COMMON SG(6000)
      DIMENSION ISG(6000)
```

FIGURE II-7-7.1. LISTING FOR SUBROUTINE RDFDBS

Reproduced from
best available copy.

```
      EQUIVALENCE (SG(1),ISG(1))
      COMMON/BLK1/FILE4(63),REC(729)
      DIMENSION IREC(729)
      EQUIVALENCE (REC(1),IREC(1))
      INTEGER FILE4
      INTEGER READER,PRINTR,OFFLIN
      DIMENSION LD1(8)
      DATA LD1(1),LD1(2),LD1(3),LD1(4)/35,154755148,152228549,431252434/
      DATA LD1(5),LD1(6),LD1(7)/756568388,756028673,755504325/
      DATA LD1(8)/801045487/
      FILE4(1)=4
      FILE4(3)=0
      FLOWT=0.0
C
C     STEP 1        OPEN THE FEED DATA BASE AND READ THE HEADER RECORD.
C
      CALL RDBASE(FILE4,REC,NREC,0)
      NSIZE=REC(1)
      NGRAV=REC(2)
      NCHAR=REC(3)
C
C     STEP 2        READ THE TOTAL NUMBER OF INPUT FEEDS.
C
      READ(READER,101) NFEED
      IF(NFEED.LE.0) GOTO 999
  101 FORMAT(3I5)
C
C     STEP 3        UNPACK FROM THE HEADER RECORD THE SPECIFICATIONS OF
C                   THE SIZE FRACTIONS, SPECIFIC GRAVITY FRACTIONS AND
C                   CHARACTERISTICS.
C
      NS=NSIZE+1
      NG=NGRAV+1
      DO 10 I=1,NS
      II=I+3
   10 SIZE(I)=REC(II)
      DO 20 J=1,NG
      JJ=J+3+NS
   20 GBOUND(J)=REC(JJ)
      JJ=JJ+1
      IASH=IREC(JJ)
      JJ=JJ+1
      ISUL=IREC(JJ)
      JJ=JJ+1
      IBTU=IREC(JJ)
      JJ=JJ+1
      ICHAR=NEXT
      IT1=NEXT+15*NCHAR
      IT2=IT1-1
```

FIGURE II-7-7.1. (Continued)

```
      DO 21 K=NEXT,IT2
      ISG(K)=IREC(JJ)
   21 JJ=JJ+1
      NEXT=IT1
      DO 1 J=1,NGRAV
    1 GRAV(J)=.5*(GBOUND(J)+GBOUND(J+1))
      IFLW=NEXT-1
      IT1=NEXT+NCHAR*NFLOWS
      IT2=IT1-1
      DO 22 K=NEXT,IT2
   22 SG(K)=0.0
      NEXT=IT1
      IDATA=NEXT
      IT1=NSIZE*NGRAV*NCHAR+NEXT
      IT2=IT1-1
      DO 30 I=NEXT,IT2
   30 SG(I)=0.0
C
C     STEP 4        INITIALIZE AT ONE THE DUMMY COUNTER USED TO CONTROL
C                   MOVEMENT THROUGH THE INPUT FEEDS.
C
      DO 40 I=1,NFEED
C
C     STEP 5        READ IN THE KEY FOR THE FEED.
C
      READ(READER,31) KEY
   31 FORMAT(A4)
C
C     STEP 6        READ THE FEED SPECIFICATION FROM THE DATA BASE.
C
      CALL RDBASE(FILE4, SG(IT1),NREC,KEY)
      IF(NREC.EQ.0) GOTO 998
C
C     STEP 7        READ IN THE FLOWRATE OF THE FEED AND ADD THIS TO THE
C                   FLOWRATE FOR THE COMBINED FEEDS.
C
      READ(READER,32) TFLOW
      FLOWT=FLOWT+TFLOW
   32 FORMAT(F10.0)
C
C     STEP 8        BLEND THE CURRENT FEED WITH THE TOTAL INPUT FEED.
C
      K=IT2
      DO 35 J=NEXT,IT2
      K=K+1
   35 SG(J)=SG(J)+SG(K)*TFLOW
C
C     STEP 9        IF THERE IS MORE THAN ONE INPUT FEED PRINT OUT THE
C                   FEED DESCRIPTION FOR THE INDIVIDUAL FEEDS.
```

FIGURE II-7-7.1. (Continued)

```
C
      IF(NFEED.EQ.1) GOTO 40
      CALL OUTPT1(SG(IT1),SIZE,GBOUND,NSIZE,NGRAV,NCHAR,      TFLOW,
     1 -I,KF,S)
C
C     STEP 10       INCREMENT BY ONE THE DUMMY COUNTER CONTROLLING
C                   MOVEMENT THROUGH THE FEEDS AND IF ALL HAVE NOT BEEN
C                   PROCRSSED LOOP BACK TO STEP 5.
C
   40 CONTINUE
C
C     STEP 11       BRANCH TO STEP 13.
C
      GOTO 999
C
C     STEP 12       THERE IS NO FEED CARRESPONDING TO THE KEY.  PRINT OUT
C                   AN ERROR MESSAGE.
C
  998 ICOL=0
      CALL PRDESC(LD1)
      CALL PRLINE
C
C     STEP 13       END LOCAL PROCESSING.
C
  999 RETURN
      END
```

FIGURE II-7-7.1. (Continued)

7.8 Subroutine SVBASE

Subroutine SVBASE writes information on a data base. Figure II-7-8.1 contains the listing and basic documentation for subroutine SVBASE.

```
      SUBROUTINE SVBASE(FILE,REC,NREC,KEY)
C
C     TITLE(CREATE A DATA BASE)
C
C     PARAMETER     DESCRIPTION
C     FILE          THIS IS A 1-DIMENSIONAL ARRAY HAVING ELEMENTS AS
C                   FOLLOWS
C                   FILE(1) IS THE NUMBER OF THE FILE ON WHICH THE DATA
C                   BASE IS TO BE WRITTEN
C                   FILE(2) IS USED TO CONTAIN THE NUMBER OF RECORDS THAT
C                   HAVE BEEN WRITTEN TO THE DATA BASE.
C                   **** FILE(2) SHOULD BE SET TO ZERO BEFORE OPENING
C                   THE FILE.****
C     REC           THIS 1-DIMENSIONAL ARRAY CONTAINS THE RECORD TO BE
C                   WRITTEN OUT TO THE DATA BASE.  WHEN THE FILE IS
C                   OPENED THIS IS THE HEADER INFORMATION.
C     NREC          THIS INTEGER VARIABLE CONTAINS THE LENGTH OF THE
C                   RECORD CURRENTLY BEING WRITTEN.
C     KEY           THIS TYPELESS VARIABLE CONTAINS THE KEY ASSOCIATED
C                   WITH THE RECORD.
C                   THE KEY SHOULD BE ZERO WHEN THE FILE IS OPENED.
C
C     VARIABLE      DESCRIPTION
C     I             THIS INTEGER VARIABLE IS USED AS A DO-LOOP COUNTER.
C     ID            THIS INTEGER VARIABLE IS USED FOR AN END-OF-FILE
C                   INDICATOR.
C     SAVE          THIS INTEGER VARIABLE IS USED FOR THE FILE NUMBER OF
C                   THE FILE CONTAINING THE DATA BASE.
C
      DIMENSION REC(NREC),FILE(2)
      INTEGER FILE,SAVE
C
C     STEP 1        BRANCH ACCORDING TO THE TYPE OF DATA BASE OPERATION
C                   THAT IS TO BE PERFORMED.
C                   BRANCH     OPERATION
C                   STEP 2     OPEN THE SAVE FILE AND WRITE THE HEADER
C                              INFORMATION.
C                   STEP 3     CLOSE THE SAVE FILE.
C                   STEP 4     WRITE A RECORD ONTO THE SAVE FILE.
C
      SAVE=FILE(1)
      IF(KEY.NE.0) GOTO 10
      IF(FILE(2).NE.0) GOTO 5
C
C     STEP 2        THE SAVE FILE IS TO BE OPENED.  WRITE THE HEADER
C                   RECORD AND END LOCAL PROCESSING.
C
      REWIND SAVE
      WRITE(SAVE) NREC,(REC(I),I=1,NREC)
```

FIGURE II-7-8.1. LISTING FOR SUBROUTINE SVBASE

```
      GOTO 900
C
C     STEP 3          THE SAVE FILE IS TO BE CLOSED.  WRITE A DOUBLE END
C                     OF FILE INDICATOR.  WRITE A PHYSICAL END-OF-FILE
C                     AND END LOCAL PROCESSING.
C
    5 ID=0
      WRITE(SAVE) ID,ID
      ENDFILE SAVE
      GOTO 900
C
C     STEP 4          A RECORD IS TO BE WRITTEN ON THE SAVE FILE.
C                     INCREMENT BY ONE THE NUMBER OF RECORDS OF THE FILE.
C                     WRITE THE RECORD KEY AND RECORD LENGTH FOLLOWED BY
C                     THE RECORD ITSELF.
C
   10 CONTINUE
      FILE(2)=FILE(2)+1
      WRITE(SAVE) KEY,NREC
      WRITE(SAVE) (REC(I),I=1,NREC)
C
C     STEP 5          END LOCAL PROCESSING.
C
  900 RETURN
      END
```

FIGURE II-7-8.1. (Continued)

7.9 Subroutine RDBASE

Subroutine RDBASE reads information from a data base. Figure II-7-9.1 contains the listing and basic documentation for subroutine RDBASE.

```
      SUBROUTINE RDBASE(FILE,REC,NREC,KEY)
C
C     TITLE(READ THE DATA BASE)
C
C     PARAMETER    DESCRIPTION
C     FILE         THIS IS A 1-DIMENSIONAL ARRAY HAVING ELEMENTS AS
C                  FOLLOWS
C                  FILE(1) IS THE NUMBER OF THE FILE ON WHICH THE DATA
C                  BASE RESIDES.
C                  FILE(2) CONTAINS THE SEQUENCE NUMBER OF THE LAST
C                  LOADED RECORD.
C                  FILE(3) CONTAINS THE NUMBER OF RECORDS ON THE DATA
C                  BASE.
C                  **** FILE(3) MUST BE SET TO ZERO BEFORE OPENING THE
C                  FILE INITIALLY****
C                  FILE(2N), FILE(2N+1), N=2,...,31 CONTAIN THE KEY
C                  AND RECORD LENGTH FOR THE (N-1)TH RECORD.
C     REC          THIS 1-DIMENSIONAL ARRAY RETURNS THE RECORD JUST READ
C     NREC         THIS INTEGER VARIABLE IS USED TO CONTAIN THE LENGTH
C                  OF THE RECORD READ.
C                  N.B. NREC=0 IF NO RECORD WITH THE SPECIFIED KEY
C                  HAS BEEN LOCATED.
C     KEY          THIS TYPELESS VARIABLE CONTAINS THE KEY ASSOCIATED
C                  WITH THE RECORD.
C                  KEY SHOULD BE SET TO ZERO WHEN OPENING THE FILE
C                  INITIALLY.
C
C     VARIABLE     DESCRIPTION
C     I            THIS INTEGER VARIABLE IS USED AS A DO-LOOP COUNTER.
C     ID           THIS INTEGER VARIABLE IS USED FOR THE DATA BASE KEYS.
C     K            THIS INTEGER VARIABLE IS USED AS AN ARRAY POINTER.
C     LENGTH       THIS INTEGER VARIABLE CONTAINS THE RECORD LENGTH.
C     LLOAD        THIS INTEGER VARIABLE CONTAINS THE SEQUENCE NUMBER
C                  OF THE LAST LOADED RECORD.
C     LOAD         THIS INTEGER VARIABLE CONTAINS THE NUMBER OF THE
C                  FILE CONTAINING THE DATA BASE.
C     NLOAD        THIS INTEGER VARIABLE CONTAINS THE NUMBER OF RECORDS
C                  IN THE DATA BASE.
C
      DIMENSION REC(NREC),FILE(63)
      INTEGER FILE
C
C     STEP 1       BRANCH ACCORDING TO THE TYPE OF DATA BASE OPERATION
C                  THAT IS TO BE PERFORMED.
C                  STEP 2   OPEN THE LOAD FILE.
C                  STEP 5   READ A RECORD FROM THE LOAD FILE.
C
      LOAD=FILE(1)
      LLOAD=FILE(2)
```

FIGURE II-7-9.1. LISTING FOR SUBROUTINE RDBASE

```
      NLOAD=FILE(3)
      IF(KEY.NE.0) GOTO 15
      IF(NLOAD.NE.0) GOTO 15
C
C     STEP 2         THE LOAD FILE IS TO BE OPENED.  READ THE HEADER
C                    RECORD.
C
      REWIND LOAD
      READ(LOAD) NREC,(REC(I),I=1,NREC)
C
C     STEP 3         READ AND SAVE THE KEYS AND LENGTHS OF ALL THE
C                    RECORDS ON THE LOAD FILE.
C
    5 READ(LOAD) ID,LENGTH
      IF(ID.EQ.0) GOTO 10
      READ(LOAD)
      NLOAD=NLOAD+1
      I=(NLOAD+1)*2
      FILE(I)=ID
      FILE(I+1)=LENGTH
      GOTO 5
C
C     STEP 4         SET THE SEQUENCE NUMBER OF THE LAST LOADED RECORD
C                    EQUAL TO THE NUMBER OF RECORDS AND BACKSPACE THE
C                    LOAD FILE SO THAT IT POINTS TO THE END OF THE LAST
C                    RECORD.  END LOCAL PROCESSING.
C
   10 LLOAD=NLOAD
      BACKSPACE LOAD
      FILE(2)=LLOAD
      FILE(3)=NLOAD
      GOTO 900
C
C     STEP 5         A RECORD IS TO BE READ FROM THE LOAD FILE.
C                    DETERMINE THE SEQUENCE NUMBER OF THE RECORD
C                    CORRESPONDING TO THE SPECIFIED KEY.
C
   15 DO 20 I=1,NLOAD
      K=(I+1)*2
      IF(KEY.EQ.FILE(K)     ) GOTO 25
   20 CONTINUE
      NREC=0
      GOTO 900
C
C     STEP 6         POSITION THE FILE AT THE BEGINNING OF THE REQUIRED
C                    RECORD.
C
   25 CONTINUE
      IF(LLOAD-I) 30,40,35
```

FIGURE II-7-9.1. (Continued)

```
   30 LLOAD=LLOAD+1
      READ(LOAD)
      IF(LLOAD.EQ.I) GOTO 45
      READ(LOAD)
      GOTO 30
   35 LLOAD=LLOAD-1
      BACKSPACE LOAD
      BACKSPACE LOAD
      GOTO 25
   40 BACKSPACE LOAD
C
C     STEP 7        READ THE RECORD
C
   45 K=(LLOAD+1)*2+1
      NREC=FILE(K)
      READ(LOAD) (REC(I),I=1,NREC)
      FILE(2)=LLOAD
C
C     STEP 8        END LOCAL PROCESSING
C
  900 RETURN
      END
```

FIGURE II-7-9.1. (Continued)

8.0 REPORT GENERATION

This chapter discusses the report generation elements. Very little time was spent during the effort which this document describes analyzing these elements. Consequently, the level of documentation for these routines is minimal. With the exception of minor format changes, these routines are exactly as they were in the original version. At various times, suggestions have been made for their enhancements such as including a cumulative percent sink in the flowstream specific gravity analysis report. To date, none of these suggestions have been tried.

8.1 The /BLK1/ and /BLK2/ Common Blocks

The purpose of the /BLK1/ and /BLK2/ common blocks is to contain the values of the unit performance measures for printing by subroutine OUTPT2. Most of these measures are computed by the unit operation routines themselves. The measures for the washing, two-stage washing, and froth flotation unit operations are calculated by subroutine CMPUS, which is presented in this chapter.

The /BLK2/ common block contains a single two-dimensional real array, DISTRB, which is used to contain the detailed distribution curve values. The contents of the /BLK1/ common block vary according to the unit operation. Figure II-8-1.1 contains a description of these storage conventions. The reader is referred to Volume I, Chapter 6, Part 4 for a detailed description of the meaning of the unit performance measures.

I Splitting

Array	Performance measure
SUMRY(ISFD)	screen analysis and characteristics of feed
SUMRY(ISCC)	screen analysis and characteristics of upper output flow
SUMRY(ISRF)	screen analysis and characteristics of lower output flow

II Screening

Array	Performance measure
SUMRY(ISFD)	screen analysis and characteristics of feed
SUMRY(ISCC)	screen analysis and characteristics of overflow stream
SUMRY(ISRF)	screen analysis and characteristics of underflow stream
SYLD	weight percent, underflow to feed
BTUREC	Btu percent, underflow to feed
SFLT	percent undersized material in overflow stream
SINK	percent oversized material in underflow stream

III Washing, two-stage washing, and froth flotation

Array	Performance measure
SUMRY(ISFD)	screen analysis and characteristics of feed
SUMRY(ISCC)	screen analysis and characteristics of clean output flow
SUMRY(ISRF)	screen analysis and characteristics of refuse output flow
SUMRY(ISMD)	screen analysis and characteristics of middlings if relevant
SYLD	percent actual recovery
TYLD	percent theoretical recovery
EFFIC	percent recovery efficiency
BTUREC	percent Btu recovery
ASHERR	percent ash error
SFLT	float in refuse, percent of product
SINK	sink in clean coal, percent of product
SMISPL	total misplaced material, percent of feed
SNRGR	near gravity 0.1 material, percent of feed
SGRAV	specific gravity of separation
SPE	probable error, specific gravity
SIMP	imperfection
SEA	error area

FIGURE II-8-1.1. CONTENTS OF /BLK1/ COMMON BLOCK

IV Rotary breaking

Array	Performance measure
SUMRY(ISFD)	screen analysis and characteristics of feed
SUMRY(ISCC)	screen analysis and characteristics of overflow (refuse) stream
SUMRY(ISRF)	screen analysis and characteristics of underflow (product) stream
SYLD	percent coal product of coal feed
SFLT	percent coal in overflow in stream
SINK	percent rock in underflow stream

V Crushing

Array	Performance measure
SUMRY(ISFD)	screen analysis and characteristics of feed
SUMRY(ISCC)	screen analysis and characteristics of crushed coal

FIGURE II-8-1.1. (Continued)

8.2 Subroutine VESSEL

Subroutine VESSEL controls the placement of the unit descriptors in the various output reports. Figure II-8-2.1 contains the lisiting and basic documentation for this routine.

```
      SUBROUTINE VESSEL(U,IFORM)
C
C     DISPLAY UNIT DESCRIPTORS)
C
C     SUBROUTINE VESSEL IS USED TO CONTROL THE DISPLAY OF THE UNIT
C     DESCRIPTORS IN ONE OF THREE FORMATS AS DESCRIBED BELOW.
C
C     PARAMETER     DESCRIPTION
C     U             THIS INTEGER VARIABLE CONTAINS THE SEQUENCE NUMBER
C                   OF THE UNIT WHOSE DESCRIPTOR IS TO BE DISPLAYED.
C     IFORM         THIS INTEGER VARIABLE SPECIFIES THE FORMAT TO BE
C                   USED IN DISPLAYING THE DESCRIPTOR. IT HAS ONE OF
C                   THREE VALUES.
C                   CODE MEANING
C                    1   DISPLAY THE DESCRIPTOR ENCLOSED IN PAREN-
C                        THESES BEGINNING IN PRINT POSITION 43. DO NOT
C                        SPACE THE PRINTER.
C                    2   PRINT THE DESCRIPTOR CENTERED ON THE PAGE.
C                        SINGLE SPACE THE PRINTER.
C                    3   DISPLAY THE DESCRIPTOR ENCLOSED IN PAREN-
C                        THESES BEGINNING IN PRINT POSITION 25. DO NOT
C                        SPACE THE PRINTER
C
C     EXTERNALS     PRDESC, PRLINE
C
C     COMMON        VARIABLES USED
C     /WORK/        IUNIT
C     /CHRSET/      LETTER
C     /CODERC/      ICOL, LINE
C     //            WS
C     /IOCNT/       CARCON,JOB
C
C     VARIABLE      DESCRIPTION
C     IU            THIS INTEGER VARIABLE CONTAINS THE ADDRESS IN WORK-
C                   ING STORAGE OF THE DESCRIPTOR.
C
      COMMON/CHRSET/LETTER(47)
      COMMON/CODERC/ICOL,LCOL,LINE(150)
      COMMON/IOCNT/JOB,CARCON
      COMMON/WORK/NEXT,LAST,ICURVE,IUNIT,IDATA,ITITLE
      COMMON WS(6000)
      INTEGER CARCON,WS,U
C
C     STEP 1        OBTAIN THE ADDRESS IN WORKING STORAGE OF THE DESCRI-
C                   PTOR. IF THERE IS NO DESCRIPTOR, END LOCAL PROCESSING
C
      IU=U+IUNIT-1
      IU=WS(IU)
      IF(IU.EQ.0)GO TO 999
```

Reproduced from best available copy.

FIGURE II-8-2.1. LISTING FOR SUBROUTINE VESSEL

```
C
C     STEP 2        BRANCH ACCORDING TO THE DESIRED FORMAT.
C
      IF(IFORM-2) 10,15,20
C
C     STEP 3        DISPLAY THE DESCRIPTOR BEGINNING IN PRINT POSITION
C                   43.
C
   10 ICOL=43
      CARCON=LETTER(37)
      JOB=-1
      LINE(ICOL)=LETTER(41)
      CALL PRDESC(WS(IU))
      ICOL=ICOL+1
      LINE(ICOL)=LETTER(42)
      CALL PRLINE
      GOTO 999
C
C     STEP 4        DISPLAY THE DESCRIPTOR CENTRED ON THE PAGE.
C
   15 ICOL=0
      CALL PRDESC(WS(IU))
      JOB=0
      CARCON=LETTER(45)
      CALL PRLINE
      GOTO 999
C
C     STEP 5        DISPLAY THE DESCRIPTOR ENCLOSED IN PARENTHESES
C                   BEGINNING IN PRINT POSITION 25.
C
   20 ICOL=25
      CARCON=LETTER(37)
      JOB=-1
      LINE(ICOL)=LETTER(41)
      CALL PRDESC(WS(IU))
      ICOL=ICOL+1
      LINE(ICOL)=LETTER(42)
      CALL PRLINE
C
C     STEP 6        END LOCAL PROCESSING.
C
  999 RETURN
      END
```

FIGURE I1-8-2.1. (Continued)

8.3 Subroutine PRHEAD

Subroutine PRHEAD controls the printing of the page heading for each page of output. Figure II-8-3.1 contains the listing and basic documentation for the routine.

```
      SUBROUTINE PRHEAD
C
C     TITLE(PRINT HEADING)
C
C     SUBROUTINE PRHEAD IS USED TO PRINT A PAGE HEADING ON THE UPPER
C     LEFT-HAND CORNER OF EACH PAGE OF OUTPUT FROM THE RUN. THIS HEAD-
C     ING CONTAINS THE USER SUPPLIED RUN DESCRIPTION ALONG WITH A SE-
C     QUENTIAL PAGE NUMBER.
C
C     COMMON          VARIABLES USED
C     /CODERC/        ICOL
C     /IOCNT/         JOB, CARCON
C     /CHRSET/        LETTER
C     /WORK/          ITITLE
C     //              WS
C
C     EXTERNALS       PRDESC, CNVTIE, PRLINE
C
C     VARIABLE        DESCRIPTION
C     IPAGE           THIS INTEGER VARIABLE CONTAINS THE PAGE NUMBER. IT
C                     IS INITIALIZED AT ZERO, AND IS INCREMENTED BY ONE
C                     UPON EACH CALL TO THIS ROUTINE
C
      COMMON/CHRSET/LETTER(47)
      COMMON/CODERC/ICOL,LCOL,LINE(150)
      COMMON/IOCNT/JOB,CARCON
      COMMON/WORK/NEXT,LAST,ICURVE,IUNIT,IDATA,ITITLE
      COMMON WS(6000)
      INTEGER WS,CARCON
      DATA IPAGE/0/
C
C     STEP 1          INITIALIZE THE NUMBER OF CHARACTERS CURRENTLY IN THE
C                     OUTPUT LINE AT ZERO.
C
      ICOL=0
C
C     STEP 2          ENTER THE USER SUPPLIED RUN TITLE INTO THE OUTPUT
C                     LINE, USING SUBROUTINE  P R D E S C .
C
      CALL PRDESC(WS(ITITLE))
C
C     STEP 3          USING SUBROUTINE  C N V D I S  ENTER THE WORD  P A G E
C                     INTO THE OUTPUT LINE.
C
      CALL CNVTIE(258726637,6,-2)
C
C     STEP 4          INCREMENT THE PAGE NUMBER BY ONE.
C
      IPAGE=IPAGE+1
```

FIGURE II-8-3.1. LISTING FOR SUBROUTINE PRHEAD

```
C
C     STEP 5          USING SUBROUTINE  C N V C I S  ENTER THE PAGE NUM-
C                     BER INTO THE OUTPUT LINE.
C
      CALL CNVTIE(IPAGE,0,-1)
C
C     STEP 6          USING SUBROUTINE  P R L I N E  WRITE THE HEADING,
C                     RIGHT-JUSTIFIED WITH A PAGE EJECT. THEN END LOCAL
C                     PROCESSING.
C
      JOB=1
      CARCON=LETTER(28)
      CALL PRLINE
      CARCON=LETTER(45)
      CALL PRLINE
      RETURN
      END
```

FIGURE II-8-3.1. (Continued)

8.4 Subroutine OUTPT1

Subroutine OUTPT1 generates the specific gravity analysis for flowstreams report. Figure II-8-4.1 contain its listing.

Reproduced from best available copy.

```
      SUBROUTINE OUTPT1(FEED,SIZE,GBOUND,NSIZE,NGRAV,NCHAR,       YFLOW,
     1 IFLOW,KF,S)
C
C      TITLE(OUTPUT FLOWSTREAM SPECIFIC GRAVITY ANALYSIS)
C
C      SUBROUTINE OUTPT1 IS USED TO GENERATE THE FLOWSTREAM SPECIFIC
C      GRAVITY REPORT.
C
C      PARAMETER      DESCRIPTION
C      FEED           THIS REAL, 3-DIMENSIONAL ARRAY CONTAINS THE DETAILED
C                     DESCRIPTION OF THE FLOW STREAM.
C      SIZE           THIS REAL, 1-DIMENSIONAL ARRAY DEFINES THE SIZE
C                     FRACTION DISTRIBUTION.
C      GBOUND         THIS REAL, 1-DIMENSIONAL ARRAY DEFINES THE SPECIFIC
C                     GRAVITY DISTRIBUTION.
C      NSIZE          THIS INTEGER VARIABLE DEFINES THE NUMBER OF SIZE
C                     FRACTIONS.
C      NGRAV          THIS INTEGER VARIABLE DEFINES THE NUMBER OF SPECIFIC
C                     GRAVITY FRACTIONS.
C      NCHAR          THIS INTEGER VARIABLE DEFINES THE NUMBER OF
C                     CHARACTERISTICS.
C      YFLOW          THIS REAL VARIABLE DEFINES THE AMOUNT OF MATERIAL
C                     IN THE FLOWSTREAM.
C      IFLOW          THIS INTEGER VARIABLE DEFINES THE FLOWSTREAM FOR
C                     WHICH VALUES ARE BEING DISPLAYED. IT CAN HAVE ONE
C                     OF TWO TYPES OF SETTINGS.
C                     CODE MEANING
C                     +I    THE VALUE FOR THE I-TH FLOWSTREAM IN THE CON-
C                           FIGURATION IS BEING DISPLAYED.
C                     -I    THE VALUE FOR THE I-TH INPUT FLOWSTREAM IS BEING
C                           DISPLAYED.
C     KF              THIS 2-DIMENSIONAL, INTEGER ARRAY DEFINES THE ORIGIN
C                     AND DESTINATION OF THE FLOWSTREAM.
C     S               THIS 1-DIMENSIONAL REAL ARRAY CONTAINS THE FLOWSTREAM
C                     TYPE CODE.
C
C     COMMON          VARIABLES USED
C     /WORK/          ICHAR,IPTU
C     /CHRSET/        LETTER
C     /CODERC/        ICOL,LINE
C     /IOCNT/         CARCON,JOF
C     //              SG
C     /BLK1/          SUMMRY
C
C     EXTERNALS       CNVTIE, PFDESC, PRHEAD, PRLINE, PRSIZE, SCSIZE
C
C     VARIABLES       DESCRIPTION
C     SS              THIS INTEGER VARIABLE IS USED TO STORE THE CURRENT
C                     FLOWSTREAM TYPE CODE
```

FIGURE II-8-4.1. LISTING FOR SUBROUTINE OUTPT1

```
C     S1,S2,S3       THESE INTEGER VARIABLES CONTAIN THE POSSIBLE
C                    FLOWSTREAM TYPE CODES C, M, R.
C     EPSILN         THIS REAL VARIABLE CONTAINS THE EPSILON VALUE FOR
C                    ZERO.
C     NLPAGE         THIS INTEGER VARIABLE CONTAINS THE NUMBER OF LINES
C                    PER PAGE.
C     NC             THIS INTEGER VARIABLE CONTAINS THE NUMBER OF
C                    CHARACTERISTICS CURRENTLY BEING PROCESSED.
C     IDIR           THIS INTEGER VARIABLE CONTROLS THE FLOWSTREAM OUTPUT
C                    IN DIRECT PERCENT
C     ICUM           THIS INTEGER VARIABLE CONTROLS THE FLOWSTREAM OUTPUT
C                    IN CUMULATIVE FORM.
C     IPS            THIS INTEGER VARIABLE COUNTS THE PASSES THROUGH THE
C                    GROUPS OF CHARACTERISTICS.
C     ICH            THIS INTEGER VARIABLE CONTAINS THE CURRENT
C                    CHARACTERISTIC DESCRIPTORS POINTER.
C     IG             THIS INTEGER VARIABLE CONTAINS THE INITIAL SPECIFIC
C                    GRAVITY FRACTION.
C     LG             THIS INTEGER VARIABLE CONTAINS THE FINAL SPECIFIC
C                    GRAVITY FRACTION.
C     IC             THIS INTEGER VARIABLE CONTAINS THE INITIAL
C                    CHARACTERISTIC.
C     TOTAL          THIS REAL VARIABLE CONTAINS THE FLOWRATE OF THE
C                    FLOWSTREAM.
C     W              THIS REAL VARIABLE CONTAINS THE WEIGHT OF FEED IN A
C                    GIVEN SIZE FRACTION.
C     IPASS          THIS INTEGER VARIABLE IS USED AS AN INDICATOR FOR
C                    THE FIRST PAGE OF THE CURRENT OUTPUT.
C     IS             THIS INTEGER VARIABLE IS USED AS A SIZE FRACTION
C                    COUNTER.
C     ISTORE         THIS INTEGER VARIABLE IS USED TO STORE THE SIZE
C                    FRACTION AT THE START OF THE CURRENT PAGE.
C     I              THIS INTEGER VARIABLE IS USED AS DO LOOP COUNTER.
C     IT1            THIS INTEGER VARIABLE IS USED AS A POINTER IN
C                    WORKING STORAGE.
C     J              THIS INTEGER VARIABLE IS USED AS A DO LOOP COUNTER.
C     K              THIS INTEGER VARIABLE IS USED AS A DO LOOP COUNTER.
C     KK             THIS INTEGER VARIABLE IS USED AS A COUNTER FOR THE
C                    CURRENT CHARACTERISTIC.
C     LINES          THIS INTEGER VARIABLE IS USED AS A LINE COUNTER FOR
C                    OUTPUT.
C     NG             THIS INTEGER VARIABLE CONTAINS THE NUMBER OF SPECIFIC
C                    GRAVITY FRACTIONS.
C     NS             THIS INTEGER VARIABLE CONTAINS THE NUMBER OF SIZE
C                    FRACTIONS.
C     CUMV           THIS REAL 1-DIMENSIONAL ARRAY IS USED FOR THE
C                    CUMULATIVE FEED ANALYSIS.
C     DIRV           THIS REAL 1-DIMENSIONAL ARRAY IS USED FOR THE DIRECT
C                    PERCENT FEED ANALYSIS.
```

FIGURE II-8-4.1. (Continued)

```
C      SUMV            THIS REAL 1-DIMENSIONAL ARRAY IS USED FOR
C                      CALCULATING THE DIRECT PERCENT AND CUMULATIVE FLOWS.
C      DIRV1,YLD,      THESE REAL VARIABLES ARE USED FOR TEMPORARY
C      SUMV1           STORAGE.
C      LD1,LD2,LD3     THESE 1-DIMENSIONAL INTEGER ARRAYS CONTAIN INTERNAL
C      LD4,LD5,LD6,    DISPLAY CODES FOR OUTPUT PURPOSES.
C      LD7,LD10,LD11
C      LD12,LD13,LSP
C      LCO,LIS,LFT,
C      LFS,LUN,LCP,
C      LMP,LRP
C
      COMMON/WORK/NEXT,LAST,ICURVE,IUNIT,IDATA,ITITLE,ICHAR,IFLW,IASH,
     1 ISUL,IRTU
      COMMON/CHRSET/LETTER(47)
      COMMON/CODERC/ICOL,LCOL,LINE(27),CA(123)
      DIMENSION DIRV(10),SUMV(10),CUMV(10)
      COMMON/BLK1/ISM(4),PM(24,13),SUMMRY(24,40)
      COMMON/IOCNT/JOB,CARCON
      COMMON SG(6000)
      INTEGER SG
      DIMENSION FEED(NSIZE,NGRAV,4),SIZE(23),GBOUND(11),KF(2,1),S(1)
      DIMENSION LSP(7),LIS(5),LFS(5),LCO(4),LFT(3),LUN(4),LCP(6),LMP(6)
     1          ,LRP(5)
      DIMENSION LD1(5),LD2(6),LD3(3),LD4(4),LD5(5),LD6(6),LD7(3),
     1       LD10(3),LD11(3),LD12(4),LD13(3)
      INTEGER CARCON
      INTEGER S,SS,S1,S2,S3
      DATA S1,S2,S3/1HC,1HM,1HR/
      DATA EPSILN/0.000000001/
      DATA NLPAGE/49/
      DATA LSP(1),LSP(2),LSP(3),LSP(4)/28,322982089,103037767,302342740/
      DATA LSP(5),LSP(6),LSP(7)/431231873,207958611,185266/
      DATA LIS(1),LIS(2),LIS(3),LIS(4)/19,154731660,760038533,20370325/
      DATA LIS(5)/ 3416402/
      DATA LFS(1),LFS(2).LFS(3),LFS(4)/17,103871955,340263469,756731586/
      DATA LFS(5)/338/
      DATA LCO(1),LCO(2),LCO(3),LCO(4)/11,688714644,154751300,42/
      DATA LFT(1),LFT(2),LFT(3)/7,101994797,1295/
      DATA LUN(1),LUN(2).LUN(3),LUN(4)/11,356029741,240439429,19/
      DATA LCP(1),LCP(2),LCP(3),LCP(4)/23,53497934,755822668,759243716/
      DATA LCP(5),LCP(6)/353192774,74701/
      DATA LMP(1),LMP(2),LMP(3),LMP(4)/22,220479756,154694893,273215765/
      DATA LMP(5),LMP(6)/55759250,973/
      DATA LRP(1),LRP(2).LRP(3),LRP(4)/19,303326547,95749263,72627501/
      DATA LRP(5)/1647565/
      DATA LD1(1),LD1(2),LD1(3),LD1(4)/20,256414153,246836053,237325134/
      DATA LD1(5)/355737338/
      DATA LD2(1),LD2(2),LD2(3),LD2(4)/25,68496697,235225679,246836053/
```

FIGURE II-8-4.1. (Continued)

```
      DATA LD2(5),LD2(6)/237325134,355737938/
      DATA LD3(1),LD3(2),LD3(3)/8,322982089,25155/
      DATA LD4(1),LD4(2),LD4(3),LD4(4)/15,69542211,347788293,302797716/
      DATA LD5(1),LD5(2),LD5(3),LD5(4)/19,55891276,22058373,783614290/
      DATA LD5(5)/807828/
      DATA LD6(1),LD6(2),LD6(3),LD6(4)/24,321233261,105386196,154987329/
      DATA LD6(5),LD6(6)/236115397,2388500/
      DATA LD7(1),LD7(2),LD7(3)/7,122164617,1305/
      DATA LD10(1),LD10(2),LD10(3)/6,103870548,38/
      DATA LD11(1),LD11(2),LD11(3)/9,54318095,50188885/
      DATA LD12(1),LD12(2),LD12(3)/15,269820101,240309190/
      DATA LD12(4)/756597719/
      DATA LD13(1),LD13(2),LD13(3)/8,103871954,5381/
C
C     STEP1         INITIALIZE THE VARIABLES CONTROLLING THE PRINTING.
C
      NC=NCHAR
      IF(NCHAR.GT.10) NC=10
      IDAR=1
      ICUM=1
      IF(NC.GT.5) ICUM=0
      IPS=0
      IC=0
      ICH=ICHAR
C
C     STEP 2        COMPUTE THE INITIAL AND FINAL SPECIFIC GRAVITY
C                   FRACTIONS TO BE PRINTED.
C
      IG=0
      DO 41 J=1,NGRAV
      W=0.0
      DO 40 I=1,NSIZE
      W=W+FEED(I,J,1)
   40 CONTINUE
      IF(W.LT.EPSILN)GO TO 41
      LG=J
      IF(IG.EQ.0)IG=J
   41 CONTINUE
      IF(IG.EQ.0)GO TO 999
      IF(IG.GT.1)IG=IG-1
      IF(LG.LT.NGRAV)LG=LG+1
      NG=LG-IG+1
C
C     STEP 3        CALL  S D S I Z E  TO COMPUTE THE SUMMARY DATA FOR
C                   THE FLOWSTREAM.
C
      CALL SDSIZE(FEED,SUMMRY,NSIZE,NGRAV,NCHAR,TOTAL)
C
C     STEP 4        PRINT OUT THE SUMMARY DATA FOR THE FLOWSTREAM.
```

FIGURE II-8-4.1. (Continued)

```
C
      IPASS=0
      NS=NSIZE+1
      IS=0
   27 IS=IS+1
      IF(IPASS.EQ.0)GO TO 28
      IF(IS.GT.NS) GOTO 18
      IF(LINES+NGRAV.LT.NLPAGE)GO TO 17
   18 IF(NCHAR.LE.5) GOTO 28
      IPS=IPS+1
      NC=NCHAR-NC-(IPS-1)*10
      IF(NC.EQ.0) GOTO 23
      IF(NC.GT.10) NC=10
      IC=IPS*10
      ICH=ICHAR+15*IC
      IS=ISTORE
      GOTO 28
   23 NC=10
      ICH=ICHAR
      IC=0
      IPS=0
      IF(NC.GT.NCHAR) NC=NCHAR
      IF(IDIR.EQ.1) GOTO 24
      ICUM=0
      IDIR=1
      GOTO 28
   24 ICUM=1
      IDIR=0
      IS=ISTORE
   28 ISTORE=IS
      IF(IS.GT.NS) GOTO 999
      LINES=0
      CALL PRHEAD
      CARCON=LETTER(45)
      JOB=0
      CALL PRDESC(LSP)
      ICOL=ICOL+1
      LINE(ICOL)=LETTER(45)
      IF(IFLOW.GT.0)GO TO 10
      CALL PRDESC(LIS)
      J=-IFLOW
      GO TO 11
   10 CALL PRDESC(LFS)
      J=IFLOW
   11 ICOL=ICOL+1
      LINE(ICOL)=LETTER(45)
      CALL CNVTIE(J,0,-1)
      IF(IPASS.EQ.0)GO TO 19
      ICOL=ICOL+1
```

FIGURE II-8-4.1. (Continued)

```
      LINE(ICOL)=LETTER(45)
      CALL PRDESC(LCO)
   19 CALL PRLINE
      CALL PRLINE
      JOB=-1
      IPASS=1
      IF(IFLOW.LE.0)GO TO 16
      IF(KF(1,IFLOW).GT.0)GO TO 12
      CALL PRDESC(LFT)
      I=2
      GO TO 22
   12 IF(KF(2,IFLOW).GT.0)GO TO 15
      SS=S(IFLOW)
      I=1
      IF(SS       .NE.S1)GO TO 13
      CALL PRDESC(LCP)
      GO TO 22
   13 IF(SS       .NE.S2)GO TO 14
      CALL PRDESC(LMP)
      GO TO 22
   14 IF(SS       .NE.S3)GO TO 15
      CALL PRDESC(LRP)
      GO TO 22
   15 CONTINUE
      ICOL=32
      CALL PRDESC(LD1)
      CALL CNVTIE(KF(1,IFLOW),4,-1)
      ICOL=ICOL+4
      LINE(ICOL)=S(IFLOW)
      ICOL=ICOL+8
      CALL PRDESC(LD2)
      CALL CNVTIE(KF(2,IFLOW),4,-1)
      CALL PRLINE
      CALL PRLINE
      GO TO 16
   22 ICOL=ICOL+1
      LINE(ICOL)=LETTER(45)
      CALL PRDESC(LUN)
      ICOL=ICOL+1
      LINE(ICOL)=LETTER(45)
      CALL CNVTIE(KF(I,IFLOW),0,-1)
      CALL PRLINE
      CALL PRLINE
   16 CONTINUE
      CALL PRLINE
      ICOL=40
      CALL PRDESC(LD3)
      IF(NCHAR.GT.5) GOTO 90
      ICOL=ICOL+12
```

FIGURE II-8-4.1. (Continued)

```
      CALL PRDESC(LD4)
      ICOL=ICOL+30
      CALL PRDESC(LD5)
      GOTO 95
 90   ICOL=ICOL+25
      IF(IDIR.EQ.1) CALL PRDESC(LD4)
      IF(ICUM.EQ.1) CALL PRDESC(LD5)
 95   CONTINUE
      CALL PRLINE
      ICOL=8
      CALL PRDESC(LD6)
      ICOL=ICOL+8
      CALL PRDESC(LD7)
      ICOL=ICOL+4
      IT1=ICH
      DO 101 I=1,NC
      CALL PRDESC(SG(IT1+9))
      ICOL=ICOL+1
101   IT1=IT1+15
      IF(NCHAR.GT.5) GOTO 112
      ICOL=ICOL+4
      IT1=ICH
      DO 102 I=1,NC
      CALL PRDESC(SG(IT1+9))
      ICOL=ICOL+1
102   IT1=IT1+15
112   CALL PRLINE
      ICOL=51
      IT1=ICH
      DO 103 I=1,NC
      CALL PRDESC(SG(IT1+12))
      ICOL=ICOL+1
103   IT1=IT1+15
      IF(NCHAR.GT.5) GOTO 114
      IT1=ICH
      ICOL=ICOL+4
      DO 104 I=1,NC
      CALL PRDESC(SG(IT1+12))
      ICOL=ICOL+1
104   IT1=IT1+15
114   CALL PRLINE
 17   IF(SUMMRY(IS,1).LT.0.0499)GO TO 51
      LINES=LINES+NG+1
      DO 74 K=1,NC
      SUMV(K)=0.0
      CUMV(K)=0.0
 74   CONTINUE
      DO 4 J=IG,LG
      IF(IS.EQ.NS)GO TO 9
```

FIGURE II-8-4.1. (Continued)

```
      DO 75 K=1,NC
      KK=K+IC
      DIRV(K)=FEED(IS,J,KK)
   75 CONTINUE
      GO TO 81
    9 DO 82 K=1,NC
      KK=K+IC
      DIRV(K)=0.0
      DO 82 I=1,NSIZE
      DIRV(K)=DIRV(K)+FEED(I,J,KK)
   82 CONTINUE
   81 IF(IC.EQ.0) DIRV1=DIRV(1)
      IF(DIRV1.LT.EPSILN ) GOTO 60
      DO 83 K=1,NC
      IF(IC.EQ.0.AND.(K.EQ.1.OR.K.EQ.IBTJ)) GOTO 83
      DIRV(K)=DIRV(K)*100.0
   83 CONTINUE
      DO 76 K=1,NC
      SUMV(K)=SUMV(K)+DIRV(K)
   76 CONTINUE
      IF(IC.EQ.0) SUMV1=SUMV(1)
      DO 79 K=1,NC
      IF(IC.EQ.0.AND.K.EQ.1) GOTO 79
      SUMV(K)=SUMV(K)/SUMV1
      DIRV(K)=DIRV(K)/DIRV1
   79 CONTINUE
      IF(IC.NE.0) GOTO 60
      YLD=10000.0/SUMMRY(IS,1)/TOTAL
      DIRV(1)=DIRV(1)*YLD
      SUMV(1)=SUMV(1)*YLD
   60 IF(J-IG-1)25,26,77
   25 IF(IS.EQ.NS)GO TO 65
      CALL PRLINE
      ICOL=3
      CALL PRSIZE(SIZE(IS),0)
      CALL CNVTIE(9837,4,-2)
      CALL PRSIZE(SIZE(IS+1),0)
      ICOL=39
      CALL PRDESC(LD10)
      GOTO 5
   65 CONTINUE
      CALL PRLINE
      ICOL=3
      CALL PRDESC(LD11)
      ICOL=ICOL+27
      CALL PRDESC(LD10)
      GOTO 5
   26 IF(IS.EQ.NS)GO TO 7
      YLD=SUMMRY(IS,1)
```

FIGURE II-8-4.1. (Continued)

```
      ICOL=3
      CALL PRDESC(LD12)
      CALL CNVTIE(YLD,6,1)
      ICOL=ICOL+15
      CALL CNVTIE(GBOUND(J),4,2)
      ICOL=ICOL+1
      LINE(ICOL)=LETTER(38)
      GOTO 5
    7 YLD=YFLOW
      ICOL=3
      CALL PRDESC(LD13)
      ICOL=ICOL+7
      CALL CNVTIE(YLD,9,1)
      ICOL=ICOL+13
      CALL CNVTIE(GBOUND(J),4,2)
      ICOL=ICOL+1
      LINE(ICOL)=LETTER(38)
      GOTO 5
   77 IF(J.EQ.LG)GO TO 78
      ICOL=40
      CALL CNVTIE(GBOUND(J),4,2)
      ICOL=ICOL+1
      LINE(ICOL)=LETTER(38)
      GOTO 5
   78 CONTINUE
      ICOL=40
      CALL CNVTIE(GBOUND(J),4,2)
      CALL CNVTIE(642552715,0,-2)
      ICOL=ICOL+1
      GOTO 6
    5 CONTINUE
      CALL CNVTIE(GBOUND(J+1),4,2)
      ICOL=ICOL+1
    6 CONTINUE
      IF(IDIR.EQ.0) GOTO 8
      DO 3 I=1,NC
      IF(IC.EQ.0.AND.I.EQ.IBTU) GOTO 2
      CALL CNVTIE(DIRV(I),8,2)
      GOTO 3
 2    CALL CNVTIE(DIRV(IBTU),8,0)
 3    CONTINUE
      ICOL=ICOL+4
    8 IF(ICUM.EQ.0) GOTO 123
      DO 121 I=1,NC
      IF(IC.EQ.0.AND.I.EQ.IBTU) GOTO 122
      CALL CNVTIE(CUMV(I),8,2)
      GOTO 121
  122 CALL CNVTIE(CUMV(IBTU),8,0)
121   CONTINUE
```

FIGURE II-8-4.1. (Continued)

```
  123 CALL PRLINE
    4 CONTINUE
   51 GOTO 27
C
C     STEP 5          END LOCAL PROCESSING.
C
  999 RETURN
      END
```

FIGURE II-8-4.1. (Continued)

8.5 Subroutine CMPUS

Subroutine CMPUS computes the unit performance measures for the washing, two-stage washing, and froth flotation unit operations. The computations themselves are described in detail in Volume I, Chapter 6, Part 4. Figure II-8-5.1 contains the listing.

```
      SUBROUTINE CMPUS(EQUIP,FEED,CC,REF,NSIZE,NGRAV,GRAV,SIZE,GBOUND
     1                ,YLDCC,YLDREF)
C
C     TITLE(COMPUTE UNIT MEASURES FOR WASHING EQUIPMENT)
C
C     SUBROUTINE CMPUS IS USED TO CONTROL THE COMUTATION OF THE UNIT
C     SPECIFIC PERFORMANCE MEASURES FOR THE UNIT OPERATIONS WHICH ARE
C     PRIMARILY DISTRIBUTION CURVE ORIENTED. THESE INCLUDE WASHING,
C     TWO-STAGE WASHING, AND FROTH FLOTATION.
C
C     PARAMETER    DESCRIPTION
C     EQUIP        THIS REAL, 1-DIMENSIONAL ARRAY CONTAINS THE EQUIP-
C                  MENT TYPE CONTROL INFORMATION. IN PARTICULAR THE
C                  FIFTH CELL OF THIS ARRAY CONTAINS THE OFFSET VALUE
C                  FOR THE DENOMINATOR OF THE CALCULATION OF INPERFEC-
C                  TION.
C     FEED         THIS REAL, 3-DIMENSIONAL ARRAY CONTAINS THE DETAILED
C                  DESCRIPTION OF THE FEED TO THE UNIT.
C     CC           THIS REAL, 3-DIMENSIONAL ARRAY CONTAINS THE DETAILED
C                  DESCRIPTION OF THE CLEAN OUTPUT FLOW FROM THE UNIT.
C     REF          THIS REAL, 3-DIMENSIONAL ARRAY CONTAINS THE DETAILED
C                  DESCRIPTION OF THE REFUSE OUTPUT FROM THE UNIT.
C     NSIZE        THIS INTEGER VARIABLE DEFINES THE NUMBER OF SIZE
C                  FRACTIONS DEFINING THE FLOWS.
C     NGRAV        THIS INTEGER VARIABLE DEFINES THE NUMBER OF SPECIFIC
C                  GRAVITY FRACTIONS DEFINING THE FLOWS.
C     GRAV         THIS REAL 1-DIMENSIONAL ARRAY CONTAINS THE MID-
C                  POINTS OF THE SPECIFIC GRAVITY FRACTION DEFINITIONS.
C     SIZE         THIS REAL, 1-DIMENSIONAL ARRAY CONTAINS THE DEFINI-
C                  TION OF THE SIZE FRACTION DISTRIBUTION.
C     GBOUND       THIS REAL, 1-DIMENSIONAL ARRAY CONTAINS THE DEFINI-
C                  TION OF THE SPECIFIC GRAVITY DISTRIBUTION.
C     YLDCC        THIS REAL VARIABLE CONTAINS THE AMOUNT OF MATERIAL
C                  IN THE CLEAN OUPTUT FLOW EXPRESSED RELATIVE TO THE
C                  WEIGHT OF THE FEED.
C     YLDREF       THIS REAL VARIABLE CONTAINS THE TOTAL AMOUNT OF
C                  MATERIAL IN THE REFUSE EXPRESSED RELATIVE TO THE
C                  TOTAL WEIGHT OF THE FEED TO THE UNIT.
C
C     COMMON       VARIABLES USED
C     /BLK1/       ISCC, ISFD, ISRF, SYLD, TYLD, EFFIC, BTUREC, ASHERR,
C                  SFLT, SINK, SMISPL, SNRGR, SGRAV, SPE, SIMP, SEA,
C                  SUMRY
C     /BLK2/       DISTRB
C     /WORK/       IASH,IBTU
C
C     EXTERNALS    YINTRP, FLOAT
C
C     VARIABLE     DESCRIPTION
```

FIGURE II-8-5.1. LISTING FOR SUBROUTINE CMPUS

Reproduced from
best available copy.

```
C     STEP 3          MOVING THROUGH ALL SIZE FRACTIONS AND THE COMPOSITE,
C                     COMPUTE THE VALUES FOR SPECIFIC GRAVITY OF SEPARA-
C                     TION, PROBABLE ERROR, IMPERFECTION, AND ACTUAL RE-
C                     COVERY.
C
      DO 411 I=1,NS
      SYLD(I)=0.0
      SIMP(I)=0.0
      SPE(I)=0.0
      SGRAV(I)=0.0
      IF(SUMRY(I,ISFD).LT.1.E-6) GOTO 411
      DO 410 J=1,NGRAV
  410 FACTOR(J)=100.0-DISTRB(I,J)
      SGRAV(I)=YINTRP(FACTOR,GRAV,50.0,NGRAV)
      SX1=YINTRP(FACTOR,GRAV,75.0,NGRAV)
      SX2=YINTRP(FACTOR,GRAV,25.0,NGRAV)
      SPE(I)=(SX1-SX2)/2.0
      IF(FACTOR(NGRAV).LT.75.0)SPE(I)=SGRAV(I)-SX2
      SIMP(I)=SPE(I)/(SGRAV(I)-EQUIP(5))
      SYLD(I)=100.0*YLDCC*SUMRY(I,ISCC)/SUMRY(I,ISFD)
  411 CONTINUE
C
C     STEP 4          MOVING THROUGH THE INDIVIDUAL SIZE FRACTIONS, COM-
C                     PUTE THE VALUES FOR THEORETICAL RECOVERY, RECOVERY
C                     EFFICIENCY, BTU RECOVERY, AND ASH ERROR.
C
      ISCA=ISCC+IASH-1
      ISCB=ISCC+IBTU-1
      ISFB=ISFD+IBTU-1
      DO 51 I=1,NSIZE
      TYLD(I)=0.0
      BTUREC(I)=0.0
      EFFIC(I)=0.0
      ASHERR(I)=0.0
      IF(SUMRY(I,ISFD).LT.1.E-6) GOTO 51
      SCMWT(1)=0.
      SCMASH(1)=0.
      WT=0.0
      DO 12 J=1,NGRAV
   12 WT=WT+FEED(I,J,1)
      DO 500 J=2,NG
      J1=J-1
      SCMWT(J)=0.
      SCMASH(J)=0.
      DO 50 K=1,J1
      SCMWT(J)=SCMWT(J)+FEED(I,K,1)
   50 SCMASH(J)=SCMASH(J)+FEED(I,K,IASH)
      IF(SCMWT(J).LT.1.E-6)GO TO 500
      SCMASH(J)=100.0*SCMASH(J)/SCMWT(J)
```

FIGURE II-8-5.1. (Continued)

```
C                       THE DISTRIBUTION CURVES.
C      K                THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                       MOVEMENT THROUGH THE INTEGERATION LOOPS WITHIN THE
C                       CALCULATION OF ERROR AREA.
C      Z                THIS DUMMY REAL VARIABLE IS USED IN THE CALCULATION
C                       OF ERROR AREA.
C      IUO, UO          THIS TYPELESS VARIABLE CONTAINS THE UNIT OPERATION
C                       TYPE CODE FOR THE UNIT.
C      ISCA,ISCB,       THESE DUMMY INTEGER VARIABLES ARE USED FOR TEMPORARY
C      ISFB             STORAGE
C
       DIMENSION FEED(NSIZE,NGRAV,5),CC(NSIZE,NGRAV,5),EQUIP(5)
      1          ,REF(NSIZE,NGRAV,5),SIZE(23),GRAV(NGRAV),GBOUND(21)
       COMMON/BLK1/ISFD,ISCC,ISRF,ISMX,SYLD(24),TYLD(24),EFFIC(24),
      1 BTUREC(24),ASHERR(24),SFLT(24),SINK(24),SMISPL(24),SNRGR(24),
      2 SGRAV(24),SPE(24),SIMP(24),SEA(24),SUMRY(24,40)
       COMMON/BLK2/DISTRB(24,20)
       COMMON/WORK/NEXT,LAST,ICURVE,IUNIT,IDATA,ITITLE,ICHAR,IFLW,IASH,
      1 ISUL,IBTU
       COMMON/LOCALV/NS,NG,SUM4,FACTOR(20),SX1,SX2,SCMWT(20),SCMASH(20)
      1                 ,J1,SG1,SG2,SG3,F,DX1,DX2,YL,YR,SUM1,SUM2,Z
      2                 ,UO,LVFILL(117)
       EQUIVALENCE (UO,IUO)
C
C      STEP 1           COMPUTE ONE PLUS THE NUMBER OF SIZE AND SPECIFIC
C                       GRAVITY FRACTIONS.
C
       NS=NSIZE+1
       NG=NGRAV+1
C
C      STEP 2           COMPUTE THE ACTUAL VALUES OF THE DISTRIBUTION CURVES
C                       FROM THE DESCRIPTIONS OF THE FEED AND CLEAN OUTPUT.
C
       UO=EQUIP(3)
   454 DO 440 J=1,NGRAV
       DISTRB(NS,J)=0.0
       SUM4=0.0
       DO 430 I=1,NSIZE
       DISTRB(NS,J)=DISTRB(NS,J)+CC(I,J,1)
       IF(IUO.NE.6)GO TO 429
       DISTRB(I,J)=0.0
       IF(FEED(I,J,1).EQ.0.0)GO TO 430
       DISTRB(I,J)=CC(I,J,1)/FEED(I,J,1)
   429 DISTRB(I,J)=DISTRB(I,J)*100.0
   430 SUM4=SUM4+FEED(I,J,1)
       IF(SUM4.LT.1.E-9)GO TO 440
       DISTRB(NS,J)=100.*DISTRB(NS,J)/SUM4
   440 CONTINUE
C
```

FIGURE II-8-5.1. (Continued)

```
C     CCWT            THIS REAL VARIABLE CONTAINS THE WEIGHT OF CLEAN COAL.
C     REFWT           THIS REAL VARIABLE CONTAINS THE WEIGHT OF REFUSE.
C     WT              THIS REAL VARIABLE CONTAINS THE WEIGHT OF FEED.
C     NS              THIS INTEGER VARIABLE CONTAINS ONE PLUS THE NUMBER
C                     OF SIZE FRACTIONS.
C     NG              THIS INTEGER VARIABLE CONTAINS ONE PLUS THE NUMBER
C                     OF SPECIFIC GRAVITY FRACTIONS.
C     J               THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                     THE MOVEMENT THROUGH SPECIFIC GRAVITY FRACTIONS.
C     I               THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                     MOVEMENT THROUGH THE SIZE FRACTIONS.
C     FACTOR          THIS REAL, 1-DIMENSIONAL ARRAY IS USED TO CONTAIN
C                     THE PERCENT TO REFUSE FOR A GIVEN DISTRIBUTION CURVE.
C     SX1             THIS REAL VARIABLE CONTAINS THAT SPECIFIC GRAVITY AT
C                     WHICH 75 PERCENT OF THE FEED WENT TO REFUSE.
C     SX2             THIS REAL VARIABLE CONTAINS THE SPECIFIC GRAVITY AT
C                     WHICH 25 PERCENT OF THE FEED WENT TO REFUSE.
C     SCMWT           THIS REAL, 1-DIMENSIONAL ARRAY CONTAINS THE CUMU-
C                     LATIVE WEIGHT BY SPECIFIC GRAVITY FRACTION FOR THE
C                     FEED.
C     SCMASH          THIS REAL, 1-DIMENSIONAL ARRAY CONTAINS THE CUMU-
C                     LATIVE WEIGHT OF ASH BY SPECIFIC GRAVITY IN THE FEED.
C     J1              THIS DUMMY INTEGER VARIABLE IS USED TO CONTAIN AN
C                     UPPER LIMIT FOR LOOPS CALCULATING CUMULATIVE VALUES
C                     FOR THE COMPOSITE.
C     SG1             THIS REAL VARIABLE CONTAINS 0.1 LESS THAN THE SPEC-
C                     IFIC GRAVITY OF SEPARATION.
C     SG2             THIS REAL VARIABLE CONTAINS THE SPECIFIC GRAVITY
C                     OF SEPARATION.
C     SG3             THIS REAL VARIABLE CONTAINS 0.1 PLUS THE SPECIFIC
C                     GRAVITY OF SEPARATION.
C     F               THIS REAL VARIABLE IS USED TO CONTAIN THE FRACTION
C                     OF A PARTICULAR SPECIFIC GRAVITY FRACTION WHICH IS
C                     WITHIN A CERTAIN RANGE.
C     DX1             THIS REAL VARIABLE CONTAINS ONE HUNDRETH OF THE
C                     DIFFERENCE BETWEEN THE SPECIFIC GRAVITY OF SEPARATION
C                     AND THE FLOAT SPECIFIC GRAVITY VALUE.
C     DX2             THIS REAL VARIABLE CONTAINS ONE HUNDRETH OF THE
C                     DIFFERENCE BETWEEN THE SINK SPECIFIC GRAVITY VALUE
C                     AND THE SPECIFIC GRAVITY OF SEPARATION.
C     YL              THIS REAL VARIABLE CONTAINS THE PERCENT TO CLEAN
C                     FOR THE FLOAT SPECIFIC GRAVITY.
C     YP              THIS REAL VARIABLE CONTAINS THE PERCENT TO CLEAN
C                     FOR THE SINK SPECIFIC GRAVITY.
C     SUM1            THIS DUMMY REAL VARIABLE IS USED IN THE COMPUTATION
C                     OF ERROR AREA.
C     SUM2            THIS REAL VARIABLE IS USED AS A DUMMY IN THE COMPUTA-
C                     TION OF ERROR AREA.
C     SUM4            THIS DUMMY VARIABLE IS USED IN THE COMPUTATION OF
```

FIGURE II-8-5.1. (Continued)

```
      IF(SCMASH(J).LT.SCMASH(J-1))SCMASH(J)=SCMASH(J-1)
      SCMWT(J)=SCMWT(J)*100.0/WT
  500 CONTINUE
      TYLD(I)=YINTRP(SCMASH,SCMWT,SUMRY(I,ISCA),NG)
      IF(TYLD(I).GT.0.)EFFIC(I)=100.*SYLD(I)/TYLD(I)
      BTUREC(I)=100.*YLDCC*SUMRY(I,ISCC)*SUMRY(I,ISCB)
     1/(SUMRY(I,ISFD)*SUMRY(I,ISFB))
      ASHERR(I)=SUMRY(I,ISCA       )-YINTRP(SCMWT,SCMASH,SYLD(I),NG)
   51 CONTINUE
C
C     STEP 5       FOR THE COMPOSITE SIZE FRACTION, COMPUTE THE VALUES
C                  FOR THEORETICAL RECOVERY, RECOVERY EFFICIENCY, BTU
C                  RECOVERY, AND ASH ERROR.
C
      SCMWT(1)=0.
      SCMASH(1)=0.
      WT=0.0
      DO 13 I=1,NSIZE
      DO 13 J=1,NGRAV
   13 WT=WT+FEED(I,J,1)
      DO 520 J=2,NG
      J1=J-1
      SCMWT(J)=0.
      SCMASH(J)=0.
      DO 52 K=1,J1
      DO 52 I=1,NSIZE
      SCMWT(J)=SCMWT(J)+100.*FEED(I,K,1)
   52 SCMASH(J)=SCMASH(J)+FEED(I,K,IASH)*100.0
      IF(SCMWT(J).LT.1.E-6)GO TO 520
      SCMASH(J)=100.*SCMASH(J)/SCMWT(J)
      IF(SCMASH(J).LT.SCMASH(J-1))SCMASH(J)=SCMASH(J-1)
      SCMWT(J)=SCMWT(J)/WT
  520 CONTINUE
      TYLD(NS)=YINTRP(SCMASH,SCMWT,SUMRY(NS,ISCA),NG)
      IF(TYLD(NS).GT.0.)EFFIC(NS)=100.*SYLD(NS)/TYLD(NS)
      BTUREC(NS)=100.*YLDCC*SUMRY(NS,ISCC)*SUMRY(NS,ISCB)/(
     1 SUMRY(NS,ISFD)*SUMRY(NS,ISFB))
      ASHERR(NS)=SUMRY(NS,ISCA       )-YINTRP(SCMWT,SCMASH,SYLD(NS),NG)
C
C     STEP 6       MOVING THROUGH THE SIZE FRACTIONS, COMPUTE THE
C                  VALUES FOR FLOAT IN REFUSE, SINK IN CLEAN COAL, TOTAL
C                  MISPLACED MATERIAL, AND NEAR GRAVITY MATERIAL.
C
   53 DO 57 I=1,NSIZE
      SFLT(I)=0.
      SINK(I)=0.
      SMISPL(I)=0.
      SNRGR(I)=0.
      IF(SUMRY(I,ISFD).LT.1.E-6) GOTO 57
```

FIGURE II-8-5.1. (Continued)

```
      SG1=SGRAV(I)-0.1
      SG2=SGRAV(I)
      SG3=SGRAV(I)+0.1
      WT=0.0
      REFWT=0.0
      CCWT=0.0
      DO 14  J=1,NGRAV
      CCWT=CCWT+CC(I,J,1)
      REFWT=REFWT+REF(I,J,1)
   14 WT=WT+FEED(I,J,1)
      DO 56  J=1,NGRAV
      IF(SG2.GT.GBOUND(J))GO TO 54
      IF(SUMRY(I,ISCC).LT.1.E-6)  GOTO 56
      SINK(I)=SINK(I)+CC(I,J,1)*100.0/CCWT
      SMISPL(I)=SMISPL(I)+100.0*CC(I,J,1)/WT
      GO TO 56
   54 IF(SG2.LT.GBOUND(J+1))GO TO 55
      IF(SUMRY(I,ISRF).LT.1.E-6)  GOTO 56
      SFLT(I)=SFLT(I)+100.0*REF(I,J,1)/REFWT
      SMISPL(I)=SMISPL(I)+100.0*REF(I,J,1)/WT
      GO TO 56
   55 F=(SG2-GBOUND(J))/(GBOUND(J+1)-GBOUND(J))
      IF(SUMRY(I,ISRF).GT.1.E-6)  SFLT(I)=SFLT(I)+F*100.*REF(I,J,1)/REFW
      IF(SUMRY(I,ISCC).GT.1.E-6)SINK(I)=SINK(I)+(1.-F)*CC(I,J,1)*100./
     1 CCWT
      SMISPL(I)=SMISPL(I)+(F*REF(I,J,1)+(1.-F)*CC(I,J,1))*100.0/WT
   56 CONTINUE
      DO 570 J=1,NGRAV
      IF((SG1.GT.GBOUND(J+1)).OR.(SG3.LT.GBOUND(J)))GO TO 570
      F=1.0
      IF((SG1.GE.GBOUND(J)).AND.(SG3.LE.GBOUND(J+1)))
     1  F=(SG3-SG1)/(GBOUND(J+1)-GBOUND(J))
      IF((SG1.LT.GBOUND(J)).AND.(SG3.LE.GBOUND(J+1)))
     1  F=(SG3-GBOUND(J))/(GBOUND(J+1)-GBOUND(J))
      IF((SG1.LT.GBOUND(J)).AND.(SG3.LE.GBOUND(J+1)))
     1 F=(SG3-GBOUND(J))/(GBOUND(J+1)-GBOUND(J))
      IF((SG1.GE.GBOUND(J)).AND.(SG3.GT.GBOUND(J+1)))
     1 F=(GBOUND(J+1)-SG1)/(GBOUND(J+1)-GBOUND(J))
      SNRGP(I)=SNRGP(I)+F*FEED(I,J,1)*100.0/WT
  570 CONTINUE
   57 CONTINUE
C
C     STEP 7          FOR THE COMPOSITE, COMPUTE THE VALUES OF SINK IN
C                     CLEAN, FLOAT IN REFUSE, AND TOTAL MISPLACED MATERIAL
C
      SG1=SGRAV(NS)-0.1
      SG2=SGRAV(NS)
      SG3=SGRAV(NS)+0.1
      SFLT(NS)=0.
```

FIGURE II-8-5.1. (Continued)

```
      SINK(NS)=0.
      SMISPL(NS)=0.
      SNRGR(NS)=0.
      WT=0.0
      CCWT=0.0
      REFWT=0.0
      DO 15 I=1,NSIZE
      DO 15 J=1,NGRAV
      CCWT=CCWT+CC(I,J,1)
      REFWT=REFWT+REF(I,J,1)
   15 WT=WT+FEED(I,J,1)
      DO 63 J=1,NGRAV
      IF(SG2.GT.GBOUND(J))GO TO 59
      DO 58 I=1,NSIZE
      SINK(NS)=SINK(NS)+100.*CC(I,J,1)/CCWT
   58 SMISPL(NS)=SMISPL(NS)+100.0*CC(I,J,1)/WT
      GO TO 63
   59 IF(SG2.LT.GBOUND(J+1))GO TO 61
      DO 60 I=1,NSIZE
      SFLT(NS)=SFLT(NS)+100.*REF(I,J,1)/REFWT
   60 SMISPL(NS)=SMISPL(NS)+100.0*REF(I,J,1)/WT
      GO TO 63
   61 F=(SG2-GBOUND(J))/(GBOUND(J+1)-GBOUND(J))
      DO 62 I=1,NSIZE
      SFLT(NS)=SFLT(NS)+100.*F*REF(I,J,1)/REFWT
      SINK(NS)=SINK(NS)+100.*F*CC(I,J,1)/CCWT
   62 SMISPL(NS)=SMISPL(NS)+100.0*(F*REF(I,J,1)+(1.-F)*CC(I,J,1))/WT
   63 CONTINUE
C
C     STEP 8         COMPUTE THE VALUE OF NEAR GRAVITY MATERIAL FOR THE
C                    COMPOSITE.
C
      DO 65 J=1,NGRAV
      IF((SG1.GT.GBOUND(J+1)).OR.(SG3.LT.GBOUND(J)))GO TO 65
      F=1.
      IF((SG1.GE.GBOUND(J)).AND.(SG3.LE.GBOUND(J+1)))
     1F=(SG3-SG1)/(GBOUND(J+1)-GBOUND(J))
      IF((SG1.LT.GBOUND(J)).AND.(SG3.LE.GBOUND(J+1)))
     1F=(SG3-GBOUND(J))/(GBOUND(J+1)-GBOUND(J))
      IF((SG1.GE.GBOUND(J)).AND.(SG3.GT.GBOUND(J+1)))
     1F=(GBOUND(J+1)-SG1)/(GBOUND(J+1)-GBOUND(J))
      DO 64 I=1,NSIZE
   64 SNRGR(NS)=SNRGR(NS)+100.*F*FEED(I,J,1)/WT
   65 CONTINUE
C
C      STEP 9        COMPUTE THE VALUE OF ERROR AREA FOR EACH SIZE FRAC-
C                    TION AND FOR THE COMPOSITE.
C
      DO 69 I=1,NS
```

FIGURE II-8-5.1. (Continued)

```
      SEA(I)=0.0
      IF(SUMRY(I,ISFD).LT.1.E-6) GOTO 69
      DX1=(SGRAV(I)-GBOUND(1))/100.0
      DX2=(GBOUND(NG)-SGRAV(I))/100.0
      DO 66 J=1,NGRAV
   66 FACTOR(J)=DISTRB(I,J)
      YL=YINTRP(GRAV,FACTOR,GBOUND(1),NGRAV)
      YR=YINTRP(GRAV,FACTOR,GBOUND(NS),NGRAV)
      SUM1=(YL+50.0)*DX1/3.0
      SUM2=(50.0+YR)*DX2/3.0
      DO 67 K=2,100,2
      Z=FLOAT(K)
      SUM1=SUM1+4.*YINTRP(GRAV,FACTOR,GBOUND(1)+(Z-1.)*DX1,NGRAV)*DX1/3.
   67 SUM2=SUM2+4.0*YINTRP(GRAV,FACTOR,SGRAV(I)+(Z-1.)*DX2,NGRAV)*DX2/3.
      DO 68 K=3,99,2
      Z=FLOAT(K)
      SUM1=SUM1+2.*YINTRP(GRAV,FACTOR,GBOUND(1)+(Z-1.)*DX1,NGRAV)*DX1/3.
   68 SUM2=SUM2+2.*YINTRP(GRAV,FACTOR,SGRAV(I)+(Z-1.)*DX2,NGRAV)*DX2/3.0
      SEA(I)=5.0*(100.0*(SGRAV(I)-GBOUND(1))-SUM1+SUM2)
   69 CONTINUE
C
C     STEP 10       END LOCAL PROCESSING.
C
  999 RETURN
      END
```

FIGURE II-8-5.1. (Continued)

8.6 Subroutine OUTPT2

Subroutine OUTPT2 generates the summary performance of units reports. Figure II-8-6.1 contains the listing.

Reproduced from
best available copy.

```
      SUBROUTINE OUTPT2(SIZE,GBOUND,NSIZE,NGRAV,NCHAR,IU,IUO,D1,D2,D3)
C
C     TITLE(REPORT ON UNIT PERFORMANCE)
C
C     SUBROUTINE OUTPT2 IS USED TO CONTROL THE PRODUCTION OF THE
C     VARIOUS REPORTS WHICH DEFINE THE PERFORMANCE OF THE UNITS WITHIN
C     THE SIMULATION.
C
C     PARAMETER     DESCRIPTION
C     SIZE          THIS 1-DIMENSIONAL REAL ARRAY CONTAINS THE SIZE
C                   DISTRIBUTION FOR THE FLOWSTREAM.
C     GBOUND        THIS REAL 1-DIMENSIONAL ARRAY CONTAINS THE DEFINITION
C                   OF THE SPECIFIC GRAVITY DISTRIBUTION FOR THE
C                   FLOWSTREAM.
C     NSIZE         THIS INTEGER VARIABLE CONTAINS THE NUMBER OF SIZE
C                   FRACTIONS.
C     NGRAV         THIS INTEGER VARIABLE CONTAINS THE NUMBER OF SPECIFIC
C                   GRAVITY FRACTIONS.
C     NCHAR         THIS INTEGER VARIABLE  CONTAINS THE NUMBER OF
C                   CHARACTERISTICS.
C     IU            THIS INTEGER VARIABLE CONTAINS THE SEQUENCE NUMBER
C                   OF THE UNIT BEING DESCRIBED.
C     IUO           THIS INTEGER VARIABLE CONTAINS THE UNIT OPERATION
C                   TYPE CODE OF THE UNIT.
C     D1            THIS REAL VARIABLE CONTAINS THE VALUE OF THE FIRST
C                   DECISION VARIABLE FOR THE UNIT.
C     D2            THIS REAL VARIABLE CONTAINS THE VALUE OF THE SECOND
C                   DECISION VARIABLE FOR THE UNIT.
C     D3            THIS REAL VARIABLE CONTAINS THE VALUE FOR THE THIRD
C                   DECISION VARIABLE FOR THE UNIT.
C
C     COMMON        VARIABLES USED
C     /BLK1/        ISM,PM,SUMRY
C     /IOCNT/       CARCON,JOB
C     /CHRSET/      LETTER
C     /BLK2/        DISTRB
C     /CODERC/      ICOL, LINE
C     /WORK/        ISTU,ICHAR
C     //            WS
C
C     EXTERNALS     FLOAT, VESSEL, PRSIZE, PRHEAD, CNVTIE, PRDESC, PRLINE
C
C     VARIABLE      DESCRIPTION
C     I,K           THESE DUMMY INTEGER VARIABLES ARE USED TO CONTROL THE
C                   MOVEMENT THROUGH LOCAL LOOPS.
C     NSTART        THIS INTEGER VARIABLE CONTAINS THE SEQUENCE NUMBER
C                   OF THE LARGEST SIZE FRACTION CONTAINING ANY MATERIAL.
C     NS            THIS INTEGER VARIABLE CONTAINS THE TOTAL NUMBER
C                   OF VALUES CONTAINED IN THE VARIOUS SUMMARY ARRAYS.
```

FIGURE II-8-6.1. LISTING FOR SUBROUTINE OUTPT2

```
C     K1             THIS INTEGER VARIABLE CONTAINS THE SEQUENCE NUMBER
C                    OF THE STARTING VALUE TO BE PRINTED ON THE CURRENT
C                    PAGE OF THE REPORT.
C     K2             THIS INTEGER VARIABLE CONTAINS THE SEQUENCE NUMBER
C                    OF THE ENDING VALUE TO BE PRINTED ON THE CURRENT PAGE
C                    OF THE REPORT.
C     IPASS          THIS INTEGER VARIABLE CONTROLS THE TITLE AND BRANCH-
C                    ING AFTER THE TITLE HAS BEEN DISPLAYED. IT HAS ONE
C                    OF THREE SETTINGS AS FOLLOWS.
C                    CODE MEANING
C                     1    PRINT TITLE FOR INITIAL PAGE OF REPORT
C                     0    PRINT TITLE FOR CONTINUATION PAGE AND BRANCH TO
C                            DISPLAY UNIT SPECIFIC MEASURES.
C                    -1    PRINT TITLE FOR CONTINUATION PAGE AND BRANCH TO
C                            DISPLAY GENERAL MEASURES.
C     K3             THIS INTEGER VARIABLE CONTAINS THE SEQUENCE NUMBER
C                    OF THE LAST SIZE FRACTION SPECIFIC VALUE ON THE
C                    CURRENT PAGE.
C     NLIM           THIS INTEGER VARIABLE CONTAINS ONE MINUS THE NUMBER
C                    OF SPECIFIC GRAVITY FRACTIONS.
C     J              THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                    MOVEMENT THROUGH THE SPECIFIC GRAVITY FRACTIONS DUR-
C                    ING THE DISPLAY OF THE WASHING DISTRIBUTION CURVES.
C     JJ,KK          THESE DUMMY INTEGER VARIABLES ARE USED AS POINTERS
C                    IN ARRAYS.
C     LD1,LD2,LD3,   THESE INTEGER ARRAYS CONTAIN PACKED CHARACTER CODES
C     LD4,LPM,NPM,   FOR DISPLAY.
C     IADD,IDIG,
C     LPMD1,LPMD2,
C     LPMD3,NGM,
C     LDOTS,
C     LDV1,LDV2,
C     LDV3,LDV4,
C     LDV5,LDV6,
C     LDV7,LDV8,
C     LDV9,LGM1,LGM
C
C     NS             THIS INTEGER VARIABLE IS ONE PLUS THE NUMBER OF
C                    SIZE FRACTIONS.
C     ISPEC          THIS INTEGER VARIABLE IS USED AS A COUNTER FOR
C                    OUTPUT PURPOSES.
C     NEW,NM,NLAB,   THESE INTEGER VARIABLES ARE USED FOR TEMPORARY
C     NN             STORAGE.
C
      DIMENSION SIZE(23),GBOUND(20)
      DIMENSION LD1(5),LD2(12),LD3(13),LD4(4)
      COMMON/WORK/NEXT,LAST,ICURVE,IUNIT,IDATA,ITITLE,ICHAR,IFLW,IASH,
     1 ISUL,IBTU
      COMMON WS(6000)
```

FIGURE II-8-6.1. (Continued)

```
      INTEGER WS
      COMMON/LOCALV/NSTART,NS,K1,K2,IPASS,K3,NLIM,LVFILL(188)
      COMMON/BLK1/ISM(4),PM(24,13),SUMRY(24,40)
      COMMON/BLK2/DISTRB(24,20)
      COMMON/CODERC/ICOL,LCOL,LINE(150)
      COMMON/IOCNT/JOB,CARCON
      COMMON/CHRSET/LETTER(47)
      INTEGER CARCON
      DIMENSION NPM(6),IADD(8),LPM(10,20),IDIG(13)
      DIMENSION LDV1(8),LDV2(3),LDV3(3),LDV4(5),LDV5(3),LDV6(5),
     11 LDV7(5),LDV8(5),LDV9(5)
      DIMENSION LGM1(6)
      DIMENSION LPMD1(9),LPMD2(7),LPMD3(3)
      DIMENSION NGM(9),LDOTS(10)
      DIMENSION LGM(7,9,4)
      DATA LD1(1),LD1(2),LD1(3),LD1(4)/17,67360793,756568388,760038533/
      DATA LD1(5)/77/
      DATA LD2(1),LD2(2),LD2(3),LD2(4)/54,324326209,308728774,760250733/
      DATA LD2(5),LD2(6),LD2(7)/273176962,303305409,118936581/
      DATA LD2(8),LD2(9),LD2(10)/303625357,20459885,253416782/
      DATA LD2(11),LD2(12)/156423061.3416402/
      DATA LD3(1),LD3(2),LD3(3),LD3(4)/57,324326209,308728774,760250733/
      DATA LD3(5),LD3(6),LD3(7)/269820303,305402755,95695361/
      DATA LD3(8),LD3(9),LD3(10)/302265605,304428297,55497670/
      DATA LD3(11),LD3(12),LD3(13)/760537684,758731586,338/
      DATA LD4(1),LD4(2),LD4(3),LD4(4)/11,688714644,154751300,42/
      DATA LPM(1,1),LPM(2,1),LPM(3,1)/43,387224008,347414612/
      DATA LPM(4,1),LPM(5,1),LPM(6,1)/155118421,235951238,205355860/
      DATA LPM(7,1),LPM(8,1),LPM(9,1)/263479621,79625199,759190659/
      DATA LPM(10,1)/21396/
      DATA LPM(1,2),LPM(2,2),LPM(3,2)/43,38686226,22057966/
      DATA LPM(4,2),LPM(5,2),LPM(6,2)/760537349,303612887,760281926/
      DATA LPM(7,2),LPM(8,2),LPM(9,2)/85216239,801045487,766958404/
      DATA LPM(10,2)/64365/
      DATA LPM(1,3),LPM(2,3),LPM(3,3)/43,356008274,321233261/
      DATA LPM(4,3),LPM(5,3),LPM(6,3)/218448210,151309129,246740357/
      DATA LPM(7,3),LPM(8,3),LPM(9,3)/303612887,760038533,20380484/
      DATA LPM(10,3)/64365/
      DATA LPM(1,4),LPM(2,4),LPM(3,4)/43,257447059,157834061/
      DATA LPM(4,4),LPM(5,4),LPM(6,4)/22041737,20107854,760537349/
      DATA LPM(7,4),LPM(8,4).LPM(9,4)/303612887,760038533,20380484/
      DATA LPM(10,4)/64365/
      DATA LPM(1,5),LPM(2,5),LPM(3,5)/43,17646913,213197123/
      DATA LPM(4,5),LPM(5,5),LPM(6,5)/257447065,801045487,801045487/
      DATA LPM(7,5),LPM(8,5),LPM(9,5)/801045487,801045487,759190659/
      DATA LPM(10,5)/21396/
      DATA LPM(1,6),LPM(2,6),LPM(3,6)/43,337662930,89166017/
      DATA LPM(4,6),LPM(5,6),LPM(6,6)/213197123,257447065,801045487/
      DATA LPM(7,6),LPM(8,6),LPM(9,6)/801045487,801045487,766958404/
```

FIGURE II-8-6.1. (Continued)

```
DATA LPM(10,6)/64365/
DATA LPM(1,7),LPM(2,7),LPM(3,7)/43,303313678,86709957/
DATA LPM(4,7),LPM(5,7),LPM(6,7)/102273225,57570031,801045487/
DATA LPM(7,7),LPM(8,7),LPM(9,7)/801045487,801045487,766958404/
DATA LPM(10,7)/64365/
DATA LPM(1,8),LPM(2,8),LPM(3,8)/43,38886226,34735365/
DATA LPM(4,8),LPM(5,8),LPM(6,8)/308739055,801045487,801045487/
DATA LPM(7,8),LPM(8,8),LPM(9,8)/801045487,801045487,766958404/
DATA LPM(10,8)/64365/
DATA LPM(1,9),LPM(2,9),LPM(3,9)/43,21793605,306771119/
DATA LPM(4,9),LPM(5,9),LPM(6,9)/801045487,801045487,801045487/
DATA LPM(7,9),LPM(8,9),LPM(9,9)/801045487,801045487,766958404/
DATA LPM(10,9)/64365/
DATA LPM(1,10),LPM(2,10),LPM(3,10)/43,103870546,757394258/
DATA LPM(4,10),LPM(5,10),LPM(6,10)/85546181,801045487,801045485/
DATA LPM(7,10),LPM(8,10),LPM(9,10)/269820101, 240309190,759243716/
DATA LPM(10,10)/86228/
DATA LPM(1,11),LPM(2,11),LPM(3,11)/43,321184493,154849484/
DATA LPM(4,11),LPM(5,11),LPM(6,11)/84206451,251972591,801045465/
DATA LPM(7,11),LPM(8,11),LPM(9,11)/766958445,766955791,766958445/
DATA LPM(10,11)/187245/
DATA LPM(1,12),LPM(2,12),LPM(3,12)/43,339558476,758420688/
DATA LPM(4,12),LPM(5,12),LPM(6,12)/201601348,758387973,304354095/
DATA LPM(7,12),LPM(8,12),LPM(9,12)/801035269,302797716,758934342/
DATA LPM(10,12)/20804/
DATA LPM(1,13),LPM(2,13),LPM(3,13)/43,236197037,122164617/
DATA LPM(4,13),LPM(5,13),LPM(6,13)/342284015,481611860,88641612/
DATA LPM(7,13),LPM(8,13),LPM(9,13)/801037165,766955791,766958445/
DATA LPM(10,13)/187245/
DATA LPM(1,14),LPM(2,14),LPM(3,14)/43,322962089,103037767/
DATA LPM(4,14),LPM(5,14),LPM(6,14)/302342740,431285749,320143442/
DATA LPM(7,14),LPM(8,14),LPM(9,14)/22057934,801045487,801045487/
DATA LPM(10,14)/195567/
DATA LPM(1,15),LPM(2,15),LPM(3,15)/43,273215617,36723525/
DATA LPM(4,15),LPM(5,15),LPM(6,15)/306771118,760021315,152604909/
DATA LPM(7,15),LPM(8,15),LPM(9,15)/122164617,342293457,801045487/
DATA LPM(10,15)/195567/
DATA LPM(1,16),LPM(2,16),LPM(3,16)/43,154468690,101987593/
DATA LPM(4,16),LPM(5,16),LPM(6,16)/25552 3823,801045487,801045487/
DATA LPM(7,16),LPM(8,16),LPM(9,16)/801045487,801045487,801045487/
DATA LPM(10,16)/195567/
DATA LPM(1,17),LPM(2,17),LPM(3,17)/43,88679378,755310913/
DATA LPM(4,17),LPM(5,17),LPM(6,17)/801045487,801045487,801045487/
DATA LPM(7,17),LPM(8,17),LPM(9,17)/801045487,801045487,801045487/
DATA LPM(10,17)/195567/
DATA LPM(1,18),LPM(2,18),LPM(3,18)/43,54266717,273215765/
DATA LPM(4,18),LPM(5,18),LPM(6,18)/55738575,20107653,85130223/
DATA LPM(7,18),LPM(8,18),LPM(9,18)/801045487,801045487,759190659/
DATA LPM(10,18)/21396/
```

FIGURE II-8-6.1. (Continued)

```
DATA LPM(1,19),LPM(2,19),LPM(3,19)/43,54268717,154850262/
DATA LPM(4,19),LPM(5,19),LPM(6,19)/86630031,397751570,84204527/
DATA LPM(7,19),LPM(8,19),LPM(9,19)/801045487,801045487,801045316/
DATA LPM(10,19)/64365/
DATA LPM(1,20),LPM(2,20),LPM(3,20)/43,305935085,154850638/
DATA LPM(4,20),LPM(5,20),LPM(6,20)/68493708,257873108,303305583/
DATA LPM(7,20),LPM(8,20),LPM(9,20)/801045487,801045487,801045316/
DATA LPM(10,20)/64365/
DATA NPM(1),NPM(2),NPM(3),NPM(4)/0,0,1101001,1111111111111/
DATA NPM(5),NPM(6)/1111111111111,1111111111111/
DATA NPM(7),NPM(8)/1100001,0/
DATA IADD(1),IADD(2),IADD(3),IADD(4)/0,0,0,4/
DATA IADD(5),IADD(6),IADD(7),IADD(8)/4,4,17,0/
DATA IDIG(1),IDIG(2),IDIG(3),IDIG(4),IDIG(5),IDIG(6)/1,1,1,1,1,1/
DATA IDIG(7),IDIG(8),IDIG(9),IDIG(10),IDIG(11)/1,1,1,2,3/
DATA IDIG(12),IDIG(13)/3,0/
DATA LPMD1(1),LPMD1(2),LPMD1(3)/36,69547282,151606537/
DATA LPMD1(4),LPMD1(5)/255519568,86617294/
DATA LPMD1(6),LPMD1(7)/347423725,386216453/
DATA LPMD1(8),LPMD1(9)/78918593,12/
DATA LPMD2(1),LPMD2(2),LPMD2(3)/28,692912451,152604909/
DATA LPMD2(4),LPMD2(5)/122164617,342282642/
DATA LPMD2(6),LPMD2(7)/17646159,60070/
DATA LPMD3(1),LPMD3(2),LPMD3(3)/6,103870546,36/
DATA NGM(1),NGM(2),NGM(3),NGM(4)/3,3,3,3/
DATA NGM(5),NGM(6),NGM(7),NGM(8),NGM(9)/4,3,3,2,2/
DATA LDOTS(1),LDOTS(2),LDOTS(3)/41,801045487,801045487/
DATA LDOTS(4)/801045487/
DATA LDOTS(5),LDOTS(6),LDOTS(7)/801045487,801045487,801045487/
DATA LDOTS(8),LDOTS(9),LDOTS(10)/801045487,801045487,47/
DATA LDV1(1),LDV1(2),LDV1(3)         /32,322982089,103037767/
DATA LDV1(4),LDV1(5),LDV1(6)/302342740,431288749,320143442/
DATA LDV1(7),LDV1(8)/22057934,2924/
DATA LDV2(1),LDV2(2),LDV2(3)/8,202695124,35692/
DATA LDV3(1),LDV3(2),LDV3(3)/10,69473093,336931692/
DATA LDV4(1),LDV4(2),LDV4(3)/17,321233261,253416400/
DATA LDV4(4),LDV4(5)/87593863,2924/
DATA LDV5(1),LDV5(2),LDV5(3)/7,757391560,339/
DATA LDV6(1),LDV6(2),LDV6(3)/17,55137480,88790213/
DATA LDV6(4),LDV6(5)/340824967,2924/
DATA LDV7(1),LDV7(2),LDV7(3)/19,356581714,756597719/
DATA LDV7(4),LDV7(5)/760038533,318316/
DATA LDV8(1),LDV8(2),LDV8(3)/16,759190659,87640911/
DATA LDV8(4),LDV8(5)/112484677,4/
DATA LDV9(1),LDV9(2),LDV9(3)/20,321233262,757391560/
DATA LDV9(4),LDV9(5)/89052114,758404296/
DATA LGM1(1),  LGM1(2),  LGM1(3)/  25,319627589,246682497/
DATA LGM1(4),  LGM1(5)  /207956611,783614290/
DATA LGM1(6)  /51701030/
```

FIGURE II-8-6.1. (Continued)

```
      DATA LGM(1,2,1),LGM(2,2,1)/4,1593668/
      DATA LGM(1,4,1),LGM(2,4,1)/4,1593668/
      DATA LGM(1,5,1),LGM(2,5,1)/4,1593668/
      DATA LGM(1,6,1),LGM(2,6,1)/4,1593668/
      DATA LGM(1,8,1),LGM(2,8,1)/4,1593668/
      DATA LGM(1,3,1),LGM(2,3,1)/4,1593668/
      DATA LGM(1,9,1),LGM(2,9,1)/4,1593668/
      DATA LGM(1,4,2),LGM(2,4,2),LGM(3,4,2)/10,53497934,755822668/
      DATA LGM(1,5,2),LGM(2,5,2),LGM(3,5,2)/10,53497934,755822668/
      DATA LGM(1,6,2),LGM(2,6,2),LGM(3,6,2)/10,53497934,755822668/
      DATA LGM(1,4,3),LGM(2,4,3),LGM(3,4,3)/6,303326547,5/
      DATA LGM(1,5,3),LGM(2,5,3),LGM(3,5,3)/6,303326547,5/
      DATA LGM(1,6,3),LGM(2,6,3),LGM(3,6,3)/6,303326547,5/
      DATA LGM(1,5,4),LGM(2,5,4),LGM(3,5,4)/9,220479756,2417107/
      DATA LGM(1,7,1),LGM(2,7,1),LGM(3,7,1)/18,101994797,680713804/
      DATA LGM(4,7,1),LGM(5,7,1)/764859535,13034/
      DATA LGM(1,7,3),LGM(2,7,3),LGM(3,7,3)/24,257447046,205355881/
      DATA LGM(4,7,3),LGM(5,7,3)/303326547,95081684/
      DATA LGM(6,7,3)/4739149/
      DATA LGM(1,7,2),LGM(2,7,2),LGM(3,7,2)/26,356008274,103871981/
      DATA LGM(4,7,2),LGM(5,7,2)/692134852,353192621/
      DATA LGM(6,7,2),LGM(7,7,2)/324084033,13/
      DATA LGM(1,8,2),LGM(2,8,2),LGM(3,8,2)/12,55137480,85119183/
      DATA LGM(4,8,2)/76/
      DATA LGM(1,9,2),LGM(2,9,2),LGM(3,9,2)/11,38335173,246702405/
      DATA LGM(4,9,2)/4/
      DATA LGM(1,3,2),LGM(2,3,2),LGM(3,3,2)/24,257447046,205355881/
      DATA LGM(4,3,2),LGM(5,3,2)/54269075,95081684/
      DATA LGM(6,3,2)/4739149/
      DATA LGM(1,3,3),LGM(2,3,3),LGM(3,3,3)/23,356008274,103871981/
      DATA LGM(4,3,3),LGM(5,3,3)/689476465,716518674/
      DATA LGM(6,3,3)/20557/
      DATA LGM(1,1,2),LGM(2,1,2),LGM(3,1,2)/6,101994797,28/
      DATA LGM(1,1,3),LGM(2,1,3),LGM(3,1,3)/6,101994797,29/
      DATA LGM(1,1,1),LGM(2,1,1),LGM(3,1,1)/7,273215765,212/
      DATA LGM(1,2,2),LGM(2,2,2),LGM(3,2,2)/24,273215765,55760685/
      DATA LGM(4,2,2),LGM(5,2,2)/693437445,313865490/
      DATA LGM(6,2,2)/1315690/
      DATA LGM(1,2,3),LGM(2,2,3),LGM(3,2,3)/24,273215765,55760749/
      DATA LGM(4,2,3),LGM(5,2,3)/691074501,313865490/
      DATA LGM(6,2,3)/1315690/
C
C     STEP 1       FIND THE SEQUENCE NUMBER OF THE SMALLEST SIZE FRAC-
C                  TION WHICH CONTAINS ANY MATERIAL. IF THE FLOWSTREAM
C                  IS EMPTY, DISPLAY THE APPROPRIATE MESSAGE, AND END
C                  LOCAL PROCESSING.
C
      ISPEC=0
      DO 20 I=1,NSIZE
```

FIGURE II-8-6.1. (Continued)

```
      JJ=ISM(1)
      IF(SUMRY(I,JJ     ).LT.1.01E-02*FLOAT(NGRAV)) GOTO 20
      NSTART=I
      GO TO 21
   20 CONTINUE
      CARCON=LETTER(28)
      JOB=-1
      ICOL=0
      CALL PRDESC(LD1)
      CALL PRLINE
      GO TO 999
C
C     STEP 2          DETERMINE THE TOTAL NUMBER OF VALUES IN THE SUMMARY
C                     ARRAYS. THIS IS ONE FOR EACH SIZE FRACTION PLUS ONE
C                     FOR THE COMPOSITE.
C
   21 NS=NSIZE+1
C
C     STEP 3          INITIALIZE THE VARIABLES SPECIFYING THE STRARTING
C                     AND ENDING SEQUENCE NUMBERS OF THE VALUES IN THE
C                     VARIOUS SUMMARY ARRAYS TO BE PRINTED ON THE CURRENT
C                     PAGE OF THE REPORT.
C
      K1=NSTART
      K2=NSTART+8
C
C     STEP 4          SET THE REPORT PAGE INDICATOR TO INDICATE THAT THE
C                     NEXT PAGE TO BE PRINTED IS THE FIRST PAGE OF THE
C                     CURRENT REPORT.
C
      IPASS=1
C
C     STEP 5          IF THE SEQUENCE NUMBER OF THE LAST COLUMN TO BE
C                     PRINTED ON THE CURRENT PAGE OF THE REPORT IS GREATER
C                     THAN THE TOTAL NUMBER OF VALUES, SET IT EQUAL TO THE
C                     TOTAL NUMBER.
C
    4 IF(K2.GT.NS)K2=NS
C
C     STEP 6          USING SUBROUTINE  P R H E A D  PRINT THE HEADING IN
C                     THE UPPER RIGHT-HAND-SIDE OF THE REPORT.
C
   90 CALL PRHEAD
      JOB=0
      CARCON=LETTER(45)
C
C     STEP 7          IF THE CURRENT PAGE IS NOT THE FIRST PAGE OF THE
C                     REPORT, BRANCH TO STEP 9.
C
C
```

FIGURE II-8-6.1. (Continued)

```
      IF(IPASS.NE.1)GO TO 11
C
C     STEP 8        DISPLAY THE TITLE FOR THE FIRST PAGE OF THE REPORT
C                   ON THE OUTPUT PAGE, AND BRANCH TO STEP 10.
C
      IF(IUO.EQ.0) GOTO 95
      ICOL=0
      CALL PRDESC(LD3)
      ICOL=ICOL+1
      CALL CNVTIE(IU,2,-1)
      CALL PRLINE
      CALL PRLINE
      GOTO 92
   95 ICOL=0
      CALL PRDESC(LD2)
      ICOL=ICOL+1
      CALL CNVTIE(IU,5,-1)
      CALL PRLINE
      CALL PRLINE
   92 CONTINUE
      GO TO 12
C
C     STEP 9        DISPLAY THE TITLE FOR A CONTINUING PAGE OF THE REPORT
C                   ON THE OUTPUT PAGE.
C
   11 IF(IUO.EQ.0) GOTO 93
      CALL PRDESC(LD3)
      ICOL=ICOL+1
      CALL CNVTIE(IU,2,-1)
      ICOL=ICOL+1
      CALL PRDESC(LD4)
      CALL PRLINE
      CALL PRLINE
      GOTO 94
   93 CONTINUE
      CALL PRDESC(LD2)
      ICOL=ICOL+1
      CALL CNVTIE(IU,5,-1)
      ICOL=ICOL+1
      CALL PRDESC(LD4)
      CALL PRLINE
      CALL PRLINE
   94 CONTINUE
C
C     STEP 10       USING SUBROUTINE  V E S S E L  DISPLAY THE UNIT
C                   DESCRIPTOR ON THE OUTPUT PAGE.
C
   12 CALL VESSEL(IU,2)
      JOB=-1
```

FIGURE II-8-6.1. (Continued)

```
C
C     STEP 11       DISPLAY THE VALUES OF THE DECISION VARIABLES ASSOC-
C                   IATED WITH THE UNIT.
C
      IF(IUO.EQ.0 ) GOTO 306
      GOTO(306,305,304,301,301,306,302,303),IUO
  301 ICOL=49
      CALL PRDESC(LDV1)
      CALL CNVTIE(D1,4,2)
      CALL PRLINE
      CALL PRLINE
      GOTO 306
  302 ICOL=23
      CALL PRDESC(LDV2)
      CALL CNVTIE(D1,5,2)
      CALL CNVTIE(756568404,0,-2)
      ICOL=ICOL+10
      CALL PRDESC(LDV3)
      CALL CNVTIE(D2,6,2)
      CALL CNVTIE(756568404,0,-2)
      ICOL=ICOL+10
      CALL PRDESC(LDV4)
      CALL CNVTIE(D3,4,2)
      CALL PRDESC(LDV5)
      CALL PRLINE
      CALL PRLINE
      GOTO 306
  303 ICOL=53
      CALL PRDESC(LDV6)
      CALL CNVTIE(D1,5,2)
      CALL PRDESC(LDV5)
      CALL PRLINE
      CALL PRLINE
      GOTO 306
  304 ICOL=53
      CALL PRDESC(LDV4)
      CALL CNVTIE(D1,5,2)
      CALL PRDESC(LDV5)
      CALL PRLINE
      CALL PRLINE
      GOTO 306
  305 ICOL=49
      CALL PRDESC(LDV7)
      CALL CNVTIE(D1,5,1)
      CALL PRDESC(LDV8)
      CALL PRLINE
      CALL PRLINE
  306 CONTINUE
C
```

FIGURE II-8-6.1. (Continued)

```
C     STEP 12        DISPLAY THE OVERALL DESCRIPTOR OF THE SPANNER OF
C                    THE REPORT.
C
      CALL PRLINE
      CALL PRDESC(LDV9)
      CALL PRLINE
C
C     STEP 13        INITIALIZE THE NUMBER OF CHARACTERS IN THE CURRENT
C                    CHARACTER ARRAY AT ZERO, IN PREPARATION FOR CONSTUCT-
C                    ING THE FIRST LINE OF THE SPANNER.
C
      ICOL=42
C
C     STEP 14        COMPUTE THE SEQUENCE NUMBER OF THE LAST ACTUAL SIZE
C                    FRACTION WHOSE VALUES ARE BEING DISPLAYED ON THE
C                    CURRENT PAGE OF THE REPORT.
C
      K3=K2
      IF(K3.EQ.NS)K3=K3-1
C
C     STEP 15        IF NO SIZE FRACTION SPECIFIC VALUES ARE TO BE
C                    DISPLAYED ON THE CURRENT PAGE, BRANCH TO STEP 19.
C
      IF(K3.LT.K1)GO TO 6
C
C     STEP 16        USING SUBROUTINES  P R S I Z E  AND  C N V D I S  AS
C                    NECESSARY, CONSTUCT AND PRINT THE FIRST LINE OF THE
C                    SPANNER FOR THE REPORT.
C
      DO 1 I=K1,K3
      CALL PRSIZE(SIZE(I),9)
    1 CONTINUE
      IF(K2.EQ.NS)CALL CNVTIF(848720,9,-2)
      CARCON=LETTER(37)
      CALL PRLINE
C
C     STEP 17        PRINT THE APPROPRIATE NUMBER OF  B Y  S FOR THE
C                    SECOND LINE OF THE REPORT.
C
      CARCON=LETTER(45)
      ICOL=49
      DO 123 I=K1,K3
      CALL CNVTIF(153,0,-2)
  123 ICOL=ICOL+7
      CALL PRLINE
C
C     STEP 18        USING SUBROUTINE  P R S I Z E  CONSTRUCT AND PRINT
C                    THE LAST LINE OF THE SPANNER FOR THE REPORT. THEN
C                    BRANCH TO STEP 20.
```

FIGURE II-8-6.1. (Continued)

```
C
      ICOL=42
      DO 2 I=K1,K3
      CALL PRSIZE(SIZE(I+1),9)
    2 CONTINUE
      CALL PRLINE
      GO TO 7
C
C     STEP 19       ONLY COMPOSITE VALUES ARE BEING DISPLAYED ON THE
C                   CURRENT PAGE. USING SUBROUTINE  C N V D I S  ENTER
C                   THE WORD  C O M P  IN THE SPANNER AND PRINT IT.
C
    6 CALL CNVTIE(848720,9,-2)
      CARCON=LETTER(37)
      CALL PRLINE
C
C     STEP 20       CONSTRUCT AND PRINT THE CURRENT PAGE.
C
    7 CONTINUE
      IF(ISPEC.EQ.1) GOTO 91
      CARCON=LETTER(45)
      IF(IUO.EQ.0) IUO=9
      DO 105 I=1,NCHAR
      CALL PRLINE
      IF(I.NE.1) GOTO 201
      CALL PRDESC(LGM1(1))
      GOTO 202
  201 IC=ICHAR+15*(I-1)
      CALL PRDESC(WS(IC+2))
  202 CONTINUE
      CALL PRLINE
      NN=NGM(IUO)
      DO 103 J=1,NN
      ICOL=3
      CALL PRDESC(LDOTS)
      ICOL=3
      CALL PRDESC(LGM(1,IUO,J))
      ICOL=44
      K=K1
  102 ICOL=ICOL+1
      JJ=ISM(J)-1+I
      IF(I.EQ.IGTU) GOTO 203
      CALL CNVTIE(SUMRY(K,JJ),6,2)
      GOTO 204
  203 CALL CNVTIE(SUMRY(K,JJ),6,0)
  204 CONTINUE
      ICOL=ICOL+2
      K=K+1
      IF(K.LE.K2) GOTO 102
```

FIGURE II-8-6.1. (Continued)

```
      CALL PRLINE
  103 CONTINUE
  105 CONTINUE
C
C     STEP 21       DISPLAY UNIT SPECIFIC PERFORMANCE MEASURES FOR THE
C                   UNIT OPERATION, IF ANY
C
      IF(IUO.LE.3.OR.IUO.GT.6) GOTO 15
      ISPEC=1
      IPASS=2
      GOTO 90
   91 CONTINUE
   15 CONTINUE
      CARCON=LETTER(45)
      CALL PRLINE
      JOB=-1
      KK=0
      K=0
      NM=NPM(IUO)
    5 CONTINUE
      IF(NM.EQ.0) GOTO 16
      KK=KK+1
      NEW=NM/10
      I=NM-NEW*10
      NM=NEW
      IF(I.EQ.0) GOTO 5
      K=K+1
      NLAB=K+IADD(IUO)
      CALL PRDESC(LPM(1,NLAB ))
      ICOL=ICOL+1
      J=K1
   10 ICOL=ICOL+1
      CALL CNVTIE(PM(J,KK),6,IDIG(KK))
      ICOL=ICOL+2
      J=J+1
      IF(J.LE.K2) GOTO 10
      CALL PRLINE
      GOTO 5
   16 CONTINUE
      IF(IUO.LE.3.OR.IUO.GT.6) GOTO 25
      CALL PRDESC(LPMD1)
      CALL PRLINE
      ICOL=2
      CALL PRDESC(LPMD2)
      CALL PRLINE
      ICOL=3
      CALL PRDESC(LDOTS)
      ICOL=3
      CALL PRDESC(LPMD3)
```

FIGURE II-8-6.1. (Continued)

```
      CALL CNVTIE(GBOUND(2),4,2)
      ICOL=44
      J=K1
      ICOL=ICOL+3
  171 CALL CNVTIE(DISTRB(J,1),5,1)
      ICOL=ICOL+4
      J=J+1
      IF(J.LE.K2) GOTO 171
      CALL PRLINE
      NLIM=NGRAV-1
      DO 625 J=2,NLIM
      ICOL=3
      CALL PRDESC(LDOTS)
      ICOL=3
      CALL CNVTIE(GBOUND(J),4,2)
      ICOL=ICOL+1
      LINE(ICOL)=LETTER(38)
      CALL CNVTIE(GBOUND(J+1),4,2)
      ICOL=44
      I=K1
      ICOL=ICOL+3
  173 CALL CNVTIE(DISTRB(I,J),5,1)
      ICOL=ICOL+4
      I=I+1
      IF(I.LE.K2) GOTO 173
      CALL PRLINE
  625 CONTINUE
      ICOL=3
      CALL PRDESC(LDOTS)
      ICOL=3
      CALL CNVTIE(321164486,0,-2)
      CALL CNVTIE(GBOUND(NGRAV),4,2)
      ICOL=44
      I=K1
      ICOL=ICOL+3
  117 CALL CNVTIE(DISTRB(I,NGRAV),5,1)
      ICOL=ICOL+4
      I=I+1
      IF(I.LE.K2) GOTO 117
      CALL PRLINE
   25 CONTINUE
C
C     STEP 22      IF THE LAST COLUMN DISPLAYED ON THE CURRENT PAGE
C                  WAS THE LAST COLUMN TO BE DISPLAYED, END LOCAL PRO-
C                  CESSING.
C
      IF(K2.EQ.NS)GO TO 999
C
C     STEP 23      SET THE PAGE COLUMN INDICATORS TO POINT TO THE NEXT
```

FIGURE II-8-6.1. (Continued)

```
C                            PAGE, RESET THE PAGE TITLE CONTROL VARIABLE, AND
C                            LOOP BACK TO STEP 5.
C
      K1=K2+1
      K2=K2+9
      IPASS=-1
      GO TO 4
  999 RETURN
      END
```

FIGURE II-8-6.1. (Continued)

8.7 Subroutine OUTPT4

Subroutine OUTPT4 produces the summary reports for units and flows. Figure II-8-7.1 lists this routine.

```
      SUBROUTINE OUTPT4
C
C     TITLE(OUTPUT SUMMARY FOR UNITS AND FLOWSTREAMS)
C
C     SUBROUTINE OUTPT4 PRODUCES THE TWO SIMULATION SUMMARY REPORTS.
C
C
C     EXTERNALS     PRHEAD, VESSEL, CNVTIE, PRDESC, PRLINE
C
C     COMMON        VARIABLES USED
C     /CNVTAR/      NUNITS, L, D1, D2, D3, VSLYLD, VSLBTU, NFLOWS, S,
C                   KF,NCHAR,WFLOW
C     /WORK/        NEXT,ICHAR,IFLW,IBTU,ISUL
C     /CHRSET/      LETTER
C     //            WS,SG
C     /CODERC/      ICOL,LINE
C     /IOCNT/       JOB,CARCON
C
C     VARIABLE      DESCRIPTION
C     LINES         THIS INTEGER VARIABLE CONTAINS THE NUMBER OF LINES
C                   PRINTED ON THE CURRENT PAGE.
C     IU            THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                   MOVEMENT THROUGH THE UNITS.
C     IEQUIP        THIS INTEGER VARIABLE CONTAINS THE ADDRESS IN WORK-
C                   STORAGE OF THE EQUIPMEMT INFORMATION FOR THE UNIT.
C     ID            THIS INTEGER VARIABLE CONTAINS THE EQUIPMENT IDENT-
C                   IFIER FOR THE UNIT.
C     IUO           THIS INTEGER VARIABLE CONTAINS THE UNIT OPERATION
C                   TYPE CODE FOR THE UNIT.
C     IF            THIS DUMMY INTEGER VARIABLE IS USED TO CONTROL THE
C                   MOVEMENT THROUGH THE FLOWS.
C     SUM3          THIS REAL VARIABLE CONTAINS THE PERCENT TOTAL
C                   SULFUR IN THE CURRENT FLOW
C     SUM4          THIS REAL VARIABLE CONTAINS THE BTU PER POUND IN
C                   THE CURRENT FLOW.
C     SUM5          THIS REAL VARIABLE CONTAINS THE PERCENT
C                   MOISTURE IN THE CURRENT FLOW.
C     FLOWPC        THIS REAL VARIABLE CONTAINS THE PERCENT OF FEED
C                   VALUE FOR THE CURRENT FLOW.
C     SO2           THIS REAL VARIABLE CONTAINS THE VALUE FOR POUNDS
C                   OF SO2 PER MILLION BTU FOR THE CURRENT FLOW.
C     NWS           THIS INTEGER VARIABLE CONTAINS THE NUMBER OF CELLS
C                   OF WORKING STORAGE REQUIRED FOR THE SIMULATION.
C     LD1-LD7,LF1,  THESE INTEGER ARRAYS CONTAIN PACKED CHARACTER CODES
C     LF3-LF6,      FOR DISPLAY.
C     LF8-LF11,
C     LF13-LF15,
C     LDASH
C
```

Reproduced from best available copy.

FIGURE II-8-7.1. LISTING FOR SUBROUTINE OUTPT4

```
C      I               THIS DUMMY INTEGER VARIABLE IS USED AS A DO-LOOP
C                      COUNTER.
C      IC              THIS INTEGER VARIABLE IS USED AS A POINTER IN
C                      WORKING STORAGE.
C      IF1,IF2,JF,     THESE INTEGER VARIABLES ARE USED FOR TEMPORARY
C      JF1,JF2,NC1,    STORAGE.
C      NC2,ISTOR
C
C      SUMM            THIS REAL VARIABLE IS USED FOR TEMPORARY STORAGE.
C      WATER           THIS REAL VARIABLE IS THE WATER CONTENT OF THE
C                      FLOWSTREAM.
C
       COMMON/CNTVAR/KF(2,100),SIZE(23),GBOUND(21),GRAV(20)
      1 ,WFLOW(100),S(100),D1(60),D2(60),D3(60),L(60)
      2                          ,NSIZE,NGRAV,IOUT,NUNITS,NFLOWS,NCHAR
       COMMON/WORK/NEXT,LAST,ICURVE,IUNIT,IDATA,ITITLE,ICHAR,IFLW,IASH,
      1 ISUL,IBTU
       COMMON/CHRSET/LETTER(47)
       COMMON WS(6000)
       COMMON/IOCNT/JOB,CARCON
       COMMON/CODERL/ICOL,LCOL,LINE(150)
       INTEGER CARCON
       INTEGER PRINTR,WS,S
       DIMENSION LF1(7),        LF3(3),LF4(3),LF5(4),LF6(3),        LF8(3),
      1 LF9(3),LF10(3),LF11(3)          ,LF13(4),LF14(6),LF15(6),LDASH(3)
       DIMENSION LD1(6),LD2(4),LD3(4),LD4(4),LD5(3),LD6(5),LD7(3)
       DIMENSION SG(6000)
       EQUIVALENCE (WS(1),SG(1))
       DATA LD1(1),LD1(2),LD1(3),LD1(4)/22,324326209,308728065,335991183/
       DATA LD1(5),LD1(6)/313873289,1299/
       DATA LD2(1),LD2(2),LD2(3),LD2(4)/12,38886226,84735365,1177/
       DATA LD3(1),LD3(2),LD3(3),LD3(4)/11,386220370,755253509,4/
       DATA LD4(1),LD4(2),LD4(3),LD4(4)/11,356029741,240439429,18/
       DATA LD5(1),LD5(2),LD5(3) /9,356029741,5346309/
       DATA LD6(1),LD6(2),LD6(3),LD6(4)/18,68432467,154987350,21532738/
       DATA LD6(5)/49491/
       DATA LD7(1),LD7(2),LD7(3)/9,692081795,1369386/
       DATA LF1(1),LF1(2),LF1(3),LF1(4)/26,324326209,308728065,335991183/
       DATA LF1(5),LF1(6),LF1(7)/313811727,390939781,4947/
       DATA LF3(1),LF3(2),LF3(3)/10,103871955,340283469/
       DATA LF4(1),LF4(2),LF4(3)/6,256414153,14/
       DATA LF5(1),LF5(2),LF5(3),LF5(4)/11,68498697,235225679,14/
       DATA LF6(1),LF6(2),LF6(3)/8,103871954,5381/
       DATA LF8(1),LF8(2),LF8(3)/8,201931603,63336/
       DATA LF9(1),LF9(2),LF9(3)/8,222074068,87173/
       DATA LF10(1),LF10(2),LF10(3)/6,240439429,18/
       DATA LF11(1),LF11(2),LF11(3)/8,356029741,58351/
       DATA LF13(1),LF13(2),LF13(3),LF13(4)/11,220513033,255512724,21/
       DATA LF14(1),LF14(2),LF14(3)/24,337679597,321181004/
```

FIGURE II-8-7.1. (Continued)

```
      DATA LF14(4),LF14(5),LF14(6)/22057934,759714901,2433348/
      DATA LF15(1),LF15(2),LF15(3)/25,755782412,330625453/
      DATA LF15(4),LF15(5),LF15(6)/389882569,236901588,256381381/
      DATA LDASH(1),LDASH(2),LDASH(3)/9,647653798,10119590/
C
C     STEP 1        USING SUBROUTINES  P R H E A D  AND  V E S S E L
C                   PRODUCE THE UNITS SUMMARY REPORT.
C
      LINES=51
      DO 10 IU=1,NUNITS
      IEQUIP=L(IU)
      IC=WS(IEQUIP-1)
      IUO=WS(IEQUIP+2)
      IF(LINES.LE.50)GO TO 5
      LINES=1
      CALL PRHEAD
      JOB=-1
      CARCON=LETTER(37)
      ICOL=50
      CALL PRDESC(LD1)
      CALL PRLINE
      CARCON=LETTER(27)
      ICOL=91
      CALL CNVTIE(421810948,0,-2)
      ICOL=ICOL+6
      CALL PRDESC(LD2)
      ICOL=ICOL+4
      CALL PRDESC(LD3)
      CALL PRLINE
      CARCON=LETTER(45)
      ICOL=8
      CALL PRDESC(LD4)
      ICOL=ICOL+8
      CALL PRDESC(LD5)
      ICOL=ICOL+25
      CALL PRDESC(LD6)
      ICOL=ICOL+10
      CALL PRDESC(LD7)
      ICOL=ICOL+5
      CALL PRDESC(LD7)
      ICOL=ICOL+10
      CALL CNVTIE(29709,0,-2)
      CALL PRLINE
      DO 100 I=9,19
  100 LINE(I)=LETTER(38)
      DO 101 I=28,36
  101 LINE(I)=LETTER(38)
      DO 102 I=62,79
  102 LINE(I)=LETTER(38)
```

FIGURE II-8-7.1. (Continued)

```
      DO 103 I=90,98
  103 LINE(I)=LETTER(38)
      DO 104 I=103,114
  104 LINE(I)=LETTER(38)
      DO 105 I=119,129
  105 LINE(I)=LETTER(38)
      ICOL=129
      CALL PRLINE
    5 LINES=LINES+2
      WATER=0.0
      IF(IUO.LT.4.OR.IUO.GT.6) GOTO 13
      DO 14 IF=1,NFLOWS
      IF(KF(2,IF).NE.IU) GOTO 17
      WATER=WATER-WFLOW(IF)
      IF1=IF
   17 CONTINUE
      IF(KF(1,IF).NE.IU) GOTO 14
      WATER=WATER+WFLOW(IF)
      IF(S(IF).EQ.LETTER( 3)) IF2=IF
   14 CONTINUE
      JF1=IFLW+NCHAR *(IF1-1)
      JF2=IFLW+NCHAR *(IF2-1)
      VSLYLD=SG(JF2+1)   *100.0/SG(JF1+1)
      JF1=JF1+IBTU
      JF2=JF2+IBTU
      VSLBTU=SG(JF2)/SG(JF1)*VSLYLD
      GOTO 18
   13 CONTINUE
      DO 11 IF=1,NFLOWS
      IF(KF(1,IF).EQ.IU) WATER=WATER+WFLOW(IF)
      IF(KF(2,IF).EQ.IU) WATER=WATER-WFLOW(IF)
   11 CONTINUE
   18 CONTINUE
      WATER=WATER*4.0
      CARCON=LETTER(27)
      ICOL=12
      CALL CNVTIE(IU,2,-1)
      ICOL=ICOL+7
      CALL CNVTIE(ID,2,-1)
      CALL PRLINE
      CARCON=LETTER(37)
      ICOL=62
      GOTO(301,302,302,304,305,306,307,308,12),IUO
  301 IF(D1(IU).LT.0.001) GOTO 12
      CALL CNVTIE(D1(IU),7,3)
      CALL PRLINE
      GOTO 12
  304 CALL CNVTIE(D1(IU),7,3)
      ICOL=ICOL+2
```

FIGURE II-8-7.1. (Continued)

```
      IF(D2(IU).GE.0.001) CALL CNVTIE(D2(IU),7,3)
      ICOL=94
      CALL CNVTIE(VSLYLD,5,1)
      ICOL=ICOL+11
      CALL CNVTIE(VSLBTU,5,1)
      CALL PRLINE
      GOTO 12
  305 CALL CNVTIE(D1(IU),7,3)
      IF((D2(IU)+D3(IU)).LT.0.001) GOTO 315
      ICOL=ICOL+2
      CALL CNVTIE(D2(IU),7,3)
      ICOL=ICOL+2
      CALL CNVTIE(D3(IU),7,3)
  315 ICOL=94
      CALL CNVTIE(VSLYLD,5,1)
      ICOL=ICOL+11
      CALL CNVTIE(VSLBTU,5,1)
      CALL PRLINE
      GOTO 12
  306 IF(D1(IU).GE.0.001) CALL CNVTIE(D1(IU),7,3)
      ICOL=94
      CALL CNVTIE(VSLYLD,5,1)
      ICOL=ICOL+11
      CALL CNVTIE(VSLBTU,5,1)
      CALL PRLINE
      GOTO 12
  308 CALL CNVTIE(D1(IU),7,3)
      CALL PRLINE
      GOTO 12
  302 CALL CNVTIE(D1(IU),7,3)
      ICOL=ICOL+2
      IF(D2(IU).GE.0.001) CALL CNVTIE(D2(IU),7,3)
      CALL PRLINE
      GOTO 12
  307 CALL CNVTIE(D1(IU),7,3)
      ICOL=ICOL+2
      CALL CNVTIE(D2(IU),7,3)
      ICOL=ICOL+2
      CALL CNVTIE(D3(IU),7,3)
      CALL PRLINE
   12 CONTINUE
      IF(ABS(WATER).LT.0.001) GOTO 10
      ICOL=120
      CALL CNVTIE(WATER,8,1)
      CALL PRLINE
   10 CALL VESSEL(IU,3)
C
C     STEP 2        USING SUBROUTINE  P R H E A D  PRODUCE THE FLOWSTREAM
C                   SUMMARY REPORT.
```

FIGURE II-8-7.1. (Continued)

```
C
      LINES=51
      NC2=0
      IF=0
   26 IF=IF+1
      IF(IF.GT.NFLOWS) GOTO 27
      IF(LINES.LE.50)GO TO 15
   27 LINES=5
      IF(IF.GT.NFLOWS.AND.NC2.EQ.NCHAR) GOTO 35
      CALL PRHEAD
      JOB=0
      CAPCON=LETTER(45)
      CALL PRDESC(LF1)
      CALL PRLINE
      CAPCON=LETTER(27)
      JOB=-1
      NC1=NC2+1
      IF(NC1.LE.NCHAR) GOTO 413
      NC1=1
  413 IF(NC1.NE.1) GOTO 414
      NC2=5
      IF(NC2.GT.NCHAR) NC2=NCHAR
      ICOL=51
      IC=ICHAR+15
      DO 401 I=2,NC2
      CALL PRDESC(WS(IC+9))
      IC=IC+15
  401 ICOL=ICOL+4
      GOTO 403
  414 IC=ICHAR+15*(NC1-1)
      IF=ISTOR
      NC2=NC1+7
      IF(NC2.GT.NCHAR) NC2=NCHAR
      ICOL=41
      DO 402 I=NC1,NC2
      CALL PRDESC(WS(IC+9))
      IC=IC+15
  402 ICOL=ICOL+4
  403 CALL PRLINE
      CAPCON=LETTER(45)
       ICOL=4
       CALL PRDESC(LF3)
       ICOL=ICOL+3
       CALL PRDESC(LF4)
        ICOL=ICOL+3
        CALL PFDESC(LF5)
        IF(NC1.EQ.1) GOTO 415
        ICOL=ICOL+4
        IC=ICHAR+(NC1-1)*15
```

FIGURE II-8-7.1. (Continued)

```
    DO 405 I=NC1,NC2
    CALL PRDESC(WS(IC+12))
    IC=IC+15
405 ICOL=ICOL+4
    GOTO 416
415 ICOL=ICOL+3
    CALL PRDESC(LF6)
    ICOL=ICOL+3
    IC=ICHAR+15
    DO 406 I=2,NC2
    CALL PRDESC(WS(IC+12))
    IC=IC+15
406 ICOL=ICOL+4
    ICOL=ICOL+1
    CALL PRDESC(LF8)
    ICOL=ICOL+5
    CALL PRDESC(LF9)
    ICOL=ICOL+4
    CALL CNVTIE(385220370,0,-2)
416 CALL PRLINE
    ICOL=6
    CALL PRDESC(LF10)
    ICOL=ICOL+4
    CALL PRDESC(LF11)
    ICOL=ICOL+4
    CALL PRDESC(LF11)
    IF(NC1.EQ.1) GOTO 417
    ICOL=ICOL+4
    DO 197 I=NC1,NC2
    CALL PRDESC(LD7)
197 ICOL=ICOL+2
    GOTO 418
417 ICOL=ICOL+7
    CALL CNVTIE(82952,0,-2)
    ICOL=ICOL+4
    DO 422 I=2,NC2
    IF(I.EQ.IBTU) GOTO 421
    CALL PRDESC(LD7)
    ICOL=ICOL+2
    GOTO 422
421 ICOL=ICOL+11
422 CONTINUE
    ICOL=ICOL+1
    CALL PRDESC(LF13)
    ICOL=ICOL+3
    CALL PRDESC(LD7)
    ICOL=ICOL+4
    CALL CNVTIE(29709,0,-2)
418 CALL PRLINE
```

FIGURE II-8-7.1. (Continued)

```
      DO 200 I=5,14
  200 LINE(I)=LETTER(38)
      DO 201 I=17,24
  201 LINE(I)=LETTER(38)
      DO 202 I=27,37
  202 LINE(I)=LETTER(38 )
      IF(NC1.EQ.1) GOTO 419
      ICOL=40
      DO 204 I=NC1,NC2
      CALL PRDESC(LDASH)
  204 ICOL=ICOL+2
      GOTO 420
  419 CONTINUE
      DO 203 I=41,48
  203 LINE(I)=LETTER(38)
      ICOL=50
      DO 205 I=2,NC2
      CALL PRDESC(LDASH)
  205 ICOL=ICOL+2
      IF1=ICOL+1
      IF2=IF1+10
      DO 208 I=IF1,IF2
  208 LINE(I)=LETTER(38)
      IF1=IF2+4
      IF2=IF1+8
      DO 209 I=IF1,IF2
  209 LINE(I)=LETTER(38)
      IF1=IF2+4
      IF2=IF1+5
      DO 210 I=IF1,IF2
  210 LINE(I)=LETTER(38)
      ICOL=127
  420 CALL PRLINE
      ISTOR=IF
   15 LINES=LINES+2
      CARCON=LETTER(27)
      ICOL=8
      CALL CNVTIE(IF,2,-1)
      ICOL=ICOL+7
      CALL CNVTIE(KF(1,IF),2,-1)
      ICOL=ICOL+4
      LINE(ICOL)=S(IF)
      ICOL=ICOL+7
      CALL CNVTIE(KF(2,IF),2,-1)
      IF(NC1.NE.1) GOTO 23
      ICOL=ICOL+9
      JF=IFLW+NCHAR *(IF-1)
      FLOWPC=SG(JF+1)
      CALL CNVTIE(FLOWPC,7,1)
```

FIGURE II-8-7.1. (Continued)

```
      ICOL=ICOL+5
      NC1=NC1+1
      DO 21 I=NC1,NC2
      JF1=JF+I
      SUMM=SG(JF1)
      IF(I.EQ.IBTU) GOTO 16
      CALL CNVTIE(SUMM,5,2)
      ICOL=ICOL+6
      GOTO 21
   16 CONTINUE
      ICOL=ICOL-1
      CALL CNVTIE(SUMM,6,0)
      ICOL=ICOL+6
   21 CONTINUE
      SO2=0.0
      IF(ISUL.EQ.0.OR.IBTU.EQ.0) GOTO 22
      JF1=JF+ISUL
      SUM3=SG(JF1)
      JF1=JF+IBTU
      SUM4=SG(JF1)
      IF(SUM4.LT.1.E-8) GOTO 22
      SO2=2.E+4*SUM3/SUM4
   22 CONTINUE
      CALL CNVTIE(SO2,6,2)
      ICOL=ICOL+6
      SUM5=WFLOW(IF)
      GPM=SUM5*4.0
      IF(SUM5.LT.1.E-8) GOTO 19
      SUM5=SUM5/(SUM5+FLOWPC)*100.0
   19 CONTINUE
      CALL CNVTIE(SUM5,6,2)
      ICOL=ICOL+2
      CALL CNVTIE(GPM,8,1)
      NC1=NC1-1
      GOTO 25
   23 CONTINUE
      ICOL=ICOL+5
      JF=IFLW+NCHAR *(IF-1)
      DO 24 I=NC1,NC2
      JF1=JF+I
      SUMM=SG(JF1)
      ICOL=ICOL+5
      IF(I.EQ.IBTU) GOTO 28
      CALL CNVTIE(SUMM,6,2)
      GOTO 24
   28 CALL CNVTIE(SUMM,6,0)
   24 CONTINUE
   25 CALL PRLINE
   30 GOTO 26
```

FIGURE II-8-7.1. (Continued)

```
C
C     STEP 3        PRINT THE.FOOTNOTE ON THE FLOWSTREAM REPORT WHICH
C                   SPECIFIES THE AMOUNT OF WORKING STORAGE USED.
C
   35 CONTINUE
      NWS=NEXT-1
      ICOL=3
      LINE(ICOL)=LETTER(39)
      ICOL=ICOL+5
      CALL PRDESC(LF14)
      CALL CNVTIE(NWS,6,-1)
      CALL PRDESC(LF15)
      CALL PRLINE
C
C     STEP 4        END LOCAL PROCESSING.
C
      RETURN
      END
```

FIGURE II-8-7.1. (Continued)

8.8 Subroutine PRLINE

Subroutine PRLINE prints a line of output. The listing and basic documentation for this routine are contained in Figure II-8-8.1.

```
      SUBROUTINE PRLINE
C
C     TITLE(PRINT A LINE OF OUTPUT)
C
C     SUBROUTINE PRLINE IS USED TO PRINT A LINE OF OUTPUT.
C
C     COMMON        VARIABLES USED
C     /CHRSET/      LETTER
C     /CODERC/      ICOL,LINE
C     /FILES/       PRINTR
C     /FLAGS/       LINEW
C     /IOCNT/       CARCON
C
C     VARIABLE      DESCRIPTION
C     I             THIS INTEGER VARIABLE IS USED AS A DUMMY DO-LOOP
C                   COUNTER.
C     NBLNK         THIS INTEGER IS USED TO CONTAIN THE NUMBER OF BLANKS
C                   TO BE INSERTED IN THE LINE.
C
      COMMON/CHRSET/LETTER(47)
      COMMON/CODERC/ICOL,LCOL,LINE(150)
      COMMON/FILES/READER,PRINTR,OFFLIN
      COMMON/FLAGS/LINEW
      COMMON/IOCNT/JOB,CARCON
      INTEGER CARCON,READER,PRINTR,OFFLIN
    1 FORMAT(136A1)
C
C     STEP 1        IF THE LINE IS BLANK PRINT THE CARRIAGE CONTROL
C                   CHARACTER AND BRANCH TO STEP 8.
C
      IF(ICOL.GT.0)GO TO 10
      WRITE(PRINTR,1)CARCON
      GOTO 999
   10 IF(ICOL.GT.LINEW)ICOL=LINEW
C
C     STEP 2        BRANCH ACCORDING TO THE REQUIRED POSITIONING OF THE
C                   LINE.
C
      IF(JOB)15,20,25
C
C     STEP 3        PRINT THE LINE LEFT JUSTIFIED AND BRANCH TO STEP 7.
C
   15 WRITE(PRINTR,1)CARCON,(LINE(I),I=1,ICOL)
      GO TO 35
C
C     STEP 4        COMPUTE THE NUMBER OF BLANKS REQUIRED TO CENTRE THE
C                   LINE.
C
   20 NBLNK=(LINEW-ICOL)/2
```

Reproduced from best available copy.

FIGURE II-8-8.1. LISTING FOR SUBROUTINE PRLINE

```
      GO TO 30
C
C     STEP 5         COMPUTE THE NUMBER OF BLANKS REQUIRED TO RIGHT
C                    JUSTIFY THE LINE.
C
   25 NBLNK=LINEW-ICOL
C
C     STEP 6         IF THE STARTING COLUMN IS BEYIND THE END OF THE LINE
C                    BRANCH TO STEP 3.  OTHERWISE PRINT THE LINE WITH
C                    THE REQUIRED NUMBER OF BLANKS.
C
   30 IF(NBLNK.LE.0)GO TO 15
      WRITE(PRINTR,1)CARCON,(LETTER(45),I=1,NBLNK),(LINE(I),I=1,ICOL)
C
C     STEP 7         RE-INITIALIZE LINE AND ICOL
C
   35 DO 40 I=1,ICOL
   40 LINE(I)=LETTER(45)
      ICOL=0
C
C     STEP 8         END LOCAL PROCESSING.
C
  999 RETURN
      END
```

Reproduced from best available copy.

FIGURE II-8-8.1. (Continued)

8.9 The /IOCNT/ Common Block

The /IOCNT/ common block contains the variables JOB and CARCON used for printer carriage control.

8.10 The /FLAGS/ Common Block

The /FLAGS/ common block contains the variable LINEW. This is set to the line width of the printer in the BLOCK DATA subroutine.

9.0 REFERENCE LIST FOR VOLUME II

(1) Applied Numerical Methods, Edited by B. Carnahan, H.A. Luther, and J.O. Wilkes, John Wiley and Sons, Inc., New York, (1969).

(2) Structured Programming, Edited by E.W. Dijkstra, O.J. Dahl, and C.A.R Hoarse, Academic Press, London (1972).

(3) S.R. Broadbent and T.G. Callcott, J. Inst. Fuel, 29, 524 (1956).

(4) S.R. Broadbent and T.G. Callcott, ibid., 29, 528 (1956).

(5) Formal Languages and Their Relation to Automata, Edited by J.E. Hopcroft and J.D. Ullman, Addison-Wesley, Reading, Massachusetts (1969).